SUN SEEBEYOND

eVIEW™ STUDIO USER'S GUIDE

Release 5.1.2

Sun microsystems

Part Number: 819-7462-10

Version 20061010122048

Contents

Appendix C

List of Figures

Chapter 1

Introduction

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

What's in This Chapter

1.1 About eView Studio

The Sun SeeBeyond eView™ Studio (eView Studio) 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, patients, vendors, businesses, inventory, and so on. In eView Studio, you define the data structure of the business objects to be stored and cross-referenced as well as 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.

1.2 eView Studio Features

eView Studio 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 Studio is to automate the creation of a highly configurable master index application. eView Studio 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 Studio automatically generates the components you need to implement a master index.

eView Studio provides the following features:

- **Rapid Development** - eView Studio 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 Studio automatically creates the eView Studio configuration files that define the primary attributes of the master index, including the configuration of the Enterprise Data Manager (EDM). eView Studio 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 Studio 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 Studio 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 Studio provides standard support for the Sun SeeBeyond Match Engine (SBME) and also provides the ability to plug in a custom matching algorithm of your choice.
- **Custom Java API** - eView Studio generates a Java API that is customized to the object structure you define. You can call the methods in this API in the Collaborations and Business Processes that define the transformation rules for data processed by the master index.
- **Standard Reports** - eView Studio provides a set of standard reports with each master index that can be run from a command line or from the EDM. The reports help you monitor the state of the data stored in the master index and help you identify configuration changes that might be required. You can also create custom reports using any ODBC-compliant reporting tool, SQL, or Java.

1.3 What's New in This Release

This release provides general maintenance fixes for eView Studio. For complete information about the changes included in this release, see the *Sun SeeBeyond eView Studio Release Notes*.

1.4 About This Document

This guide provides comprehensive information on using eView Studio to design, configure, and create a customized enterprise-wide master index. This guide explains how to install eView Studio 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 Studio and master index structure. This guide is intended to be used with the *Sun SeeBeyond eView Studio Configuration Guide*, *Implementing the Sun SeeBeyond Match Engine with eView Studio*, and the *Sun SeeBeyond eView Studio Reference Guide*.

1.4.1 What's in This Document

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

- **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 Studio and the master index, and describes how a master index is created.
- **Chapter 3 "Installation"** gives instructions for installing the eView Studio files and setting up the environment for eView Studio 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 Studio Project, and provides information about the XML Editor.
- **Chapter 6 "Creating Custom Plug-ins"** describes how to implement custom processing code in the master index.
- **Chapter 7 "Building the Application"** explains how to generate the eView Studio application to create a master index from the files you created and customized.
- **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 Studio 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 and deploy the Deployment Profile for the master index.
- **Chapter 12 "Maintenance Tasks"** provides information on backing up the master index and using the monitor to view logs and alerts.
- **Chapter 13 "Implementing the eView Studio Sample"** explains how to work with the eView Studio 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 "Initial Load Tips"** provides guidelines to help the initial process of loading date run smoothly.
- **Appendix C "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.

1.4.2 Scope

This guide provides step-by-step instructions for installing eView Studio and creating a master index application. It includes navigational information, functional instructions, and background information where required. This guide does not include information or instructions on using the EDM or eGate Integrator. These topics are covered in the appropriate user guide (for more information, see **"Related Documents" on page 18**).

1.4.3 Intended Audience

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

1.4.4 Text Conventions

The following conventions are observed throughout this document.

Table 1 Text Conventions

Text Convention	Used For	Examples
Bold	Names of buttons, files, icons, parameters, variables, methods, menus, and objects	▪ Click **OK**. ▪ On the **File** menu, click **Exit**. ▪ Select the **eGate.sar** file.
`Monospaced`	Command line arguments, code samples; variables are shown in ***bold italic***	`java -jar` ***filename*** `.jar`
Blue bold	Hypertext links within document	See **Text Conventions** on page 18
Blue underlined	Hypertext links for Web addresses (URLs) or email addresses	http://www.sun.com

1.4.5 Screenshots

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

1.5 Related Documents

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

- *Sun SeeBeyond Enterprise Data Manager User's Guide*
- *Sun SeeBeyond eView Studio Configuration Guide*
- *Sun SeeBeyond eView Studio Reference Guide*
- *Sun SeeBeyond eView Studio Reporting Guide*
- *Implementing the Sun SeeBeyond Match Engine with eView Studio*
- *Sun SeeBeyond eGate Integrator User's Guide*
- *Sun SeeBeyond eGate Integrator System Administration Guide*
- *Sun SeeBeyond eInsight Business Process Manager User's Guide*

1.6 Sun Microsystems, Inc. Web Site

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

http://www.sun.com

1.7 Documentation Feedback

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

CAPS_docsfeedback@sun.com

Chapter 2

eView Studio Overview

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

What's in This Chapter

2.1 About eView Studio

eView Studio is an application building tool that allows you to design, configure, and create matching and indexing applications called enterprise-wide master indexes (or just *master indexes*). The master index 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, patients, vendors, businesses, hardware parts, and so on. In eView Studio, you define the data structure of the business objects to be stored and cross-referenced as well as the logic that determines how data is updated, standardized, weighted, and matched in the master index.

2.2 eView Studio and the Java Composite Application Platform Suite

Each eView Studio application is created within the structure of an eGate Project in Enterprise Designer. eView Studio is tightly integrated within the Java Composite Application Platform Suite and can leverage the features of other components of the suite.

eGate Integrator

eView Studio 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 Studio method OTD to the external system Collaborations or Business Processes.

eInsight Business Process Manager

eInsight™ Business Process Manager (BPM) 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 Studio Project in Business Processes, enabling access to the master index database through eInsight BPM.

eVision Studio

eVision™ Studio is a graphical design studio for the WYSIWYG creation of specialized web applications, specifically developed for integration with the eGate and eInsight BPM runtime environments. Using eVision Studio, you can create custom web pages to access information stored in the master index databases.

2.3 eView Studio Repository Components

The components of eView Studio are designed to work within 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 Studio include the following.

- **eView Wizard** on page 21
- **Editors** on page 22
- **Project Components** on page 22
- **Environment Components** on page 26

2.3.1 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 Enterprise Data Manager (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.

2.3.2 Editors

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

- **XML Editor** - allows you to review and customize the XML configuration files created by the eView Wizard. Enterprise Designer provides schema validation services, while the XML Editor provides verification for XML syntax. The XML editor is automatically launched when you open an eView Studio 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 Studio database script or configuration file.
- **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.

2.3.3 Project Components

An eView Studio master index is implemented within a Project in Enterprise Designer. When you create an eView Studio application, a set of configuration files and a set of database files are generated based on the information you specified in the eView Studio Wizard. When you generate the Project, additional components are created, including a method OTD, an outbound OTD, eInsight Business Process 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 can be added to the client Projects that access the master index, including Services, Collaborations, OTDs, Web Connectors, eWays, JMS Queues, JMS Topics, Business Processes, and so on. You can use the standard Enterprise Designer editors, such as the OTD or Collaboration editors, to create these components.

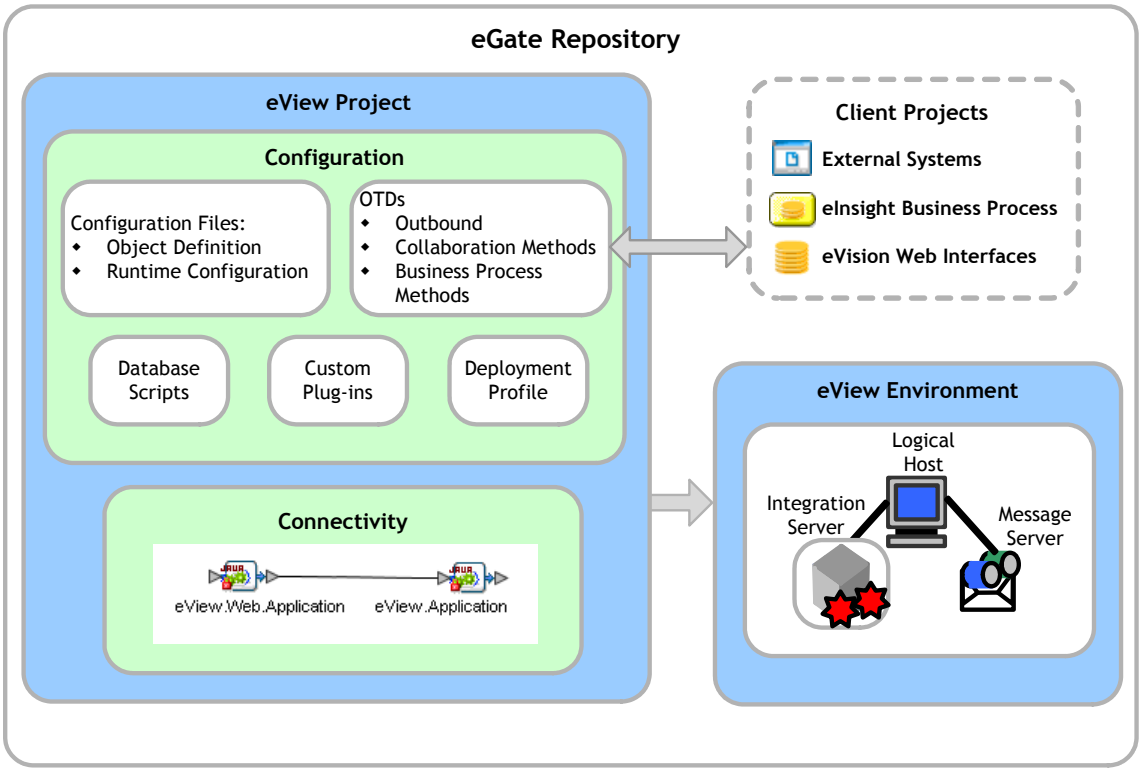
Following is a list of eView Studio Project components.

- 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 23 illustrates the Project and Environment components of eView Studio.

Figure 1 eView Studio 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 the runtime components of the master index, which are listed in "Master Index Runtime Components" on page 29.

- **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 or used 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 single best record (SBR). You can define custom update procedures in this file.
- **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 to define external systems and code lists. Additional scripts to create or drop database tables are created when you generate the Project (or by the wizard if you choose to generate all Project files in the wizard).

- **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.
- **Code List** - Contains the SQL statements to insert processing codes and drop-down list values into the database. Some of the entries in this file are generated by the wizard. Code lists must be defined 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 specified in the eView Wizard. You can customize this file and then run it against an Oracle database to create a customized master index database.
- **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 Studio Project and run against the master index database.

Custom Plug-ins

eView Studio 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

Studio 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. For example, if you create a custom Query Builder, it must be listed in the Candidate Select file to be accessible to the master index.

Match Engine Configuration Files

Several configuration files for the SBME are created in the eView Studio 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 these files as needed. For more information, refer to *Implementing the Sun SeeBeyond Match Engine with eView Studio*.

Outbound Object Type Definition (OTD)

eView Studio 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 external systems. It includes the objects and fields defined in the Object Definition file plus additional SBR information (such as the create date and create user) and additional system object information (such as the local ID and system code). If you plan to use this OTD to make the master index data available to external systems, you must define a JMS Topic in the eView Studio Connectivity Map to which the master index can publish transactions.

Dynamic Java API

Due to the flexibility of the object structure, eView Studio generates several dynamic Java methods for use in Collaborations and in Business Processes. One set is provided in a method OTD for use in Collaborations and one set is provided for Business Processes. The names, parameter types, and return types of these methods vary based on the objects you defined in the object structure. These methods are described in the *Sun SeeBeyond eView Studio Reference Guide*.

Method OTD

Generating the eView Studio application creates a method OTD that includes 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, and so on.

Business Process Java Methods

In addition to the method OTD, which can be used in Collaborations, eView Studio creates a set of Java methods that can be incorporated into eInsight Business Processes and into eVision Web Services. These methods are a subset of those defined for the method OTD, providing the ability to view, retrieve, and match information in the master index database.

Connectivity Components

The eView Studio Project Connectivity Map consists of two required components: the web application service and the application service. Two optional components are a JMS Topic for broadcasting messages and an Oracle eWay for database connectivity (the Oracle eWay is required if eView Studio runs on the Sun SeeBeyond Integration Server). 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 include those created for the external systems sharing data with the index through a Collaboration or Business Process.

For client Projects, you can use connectivity components from the eView Studio server Project and 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 or Business Processes. They can also route the processed data back to the appropriate local systems through eWays.

Deployment Profile

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

2.3.4 Environment Components

The eView Studio Environments define the deployment environment of the master index, including the Logical Host, application or integration server, external systems, and so on. If eView Studio client Projects use the same Environment, it might 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 Studio applications. You must define and configure at least one Environment for the master index before you can deploy the application. The application or integration server hosting the eView Studio application is configured within the Environment in Enterprise Designer.

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

2.4 Learning about the Master Index Runtime Environment

In today's business environment, important information about certain business objects in your organization might exist in many disparate information systems. It is vital that this information flow seamlessly and rapidly between departments and systems

throughout the entire business network. As organizations grow, merge, and form affiliations, sharing data between different information systems becomes a complicated task. The master indexes you create from eView Studio 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.

2.4.1 Functions of the Runtime Environment

In the runtime environment, 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 what information from each record to preserve in the resulting record.
- **Reports** - You can generate reports that provide information about the current state of the data in the master index, helping you monitor stored data and determine how that data needs to be updated. Report information also helps verify that the matching logic and weight thresholds are defined correctly.

2.4.2 Features of the Runtime Environment

The runtime 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-referencing 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, allowing you to maintain your current systems and practices while maintaining the most current and accurate information.
- **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 and can be configured to automatically match 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 external systems, providing seamless integration with the systems in your enterprise.
- **Matching Algorithm** - The master index is designed to use the SBME or a custom matching algorithm to provide a matching probability weight between records. The

SBME algorithm provides the flexibility to create user-defined matching thresholds, which control how potential duplicates and automatic matches are determined.

- **Shared Information** - Each time a record is updated, added, merged, or unmerged from the EDM, 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.

2.5 Master Index Runtime Components

The master indexes created by eView Studio are made up of several components that work together to form the complete indexing system. The primary components of the master index consists of the following.

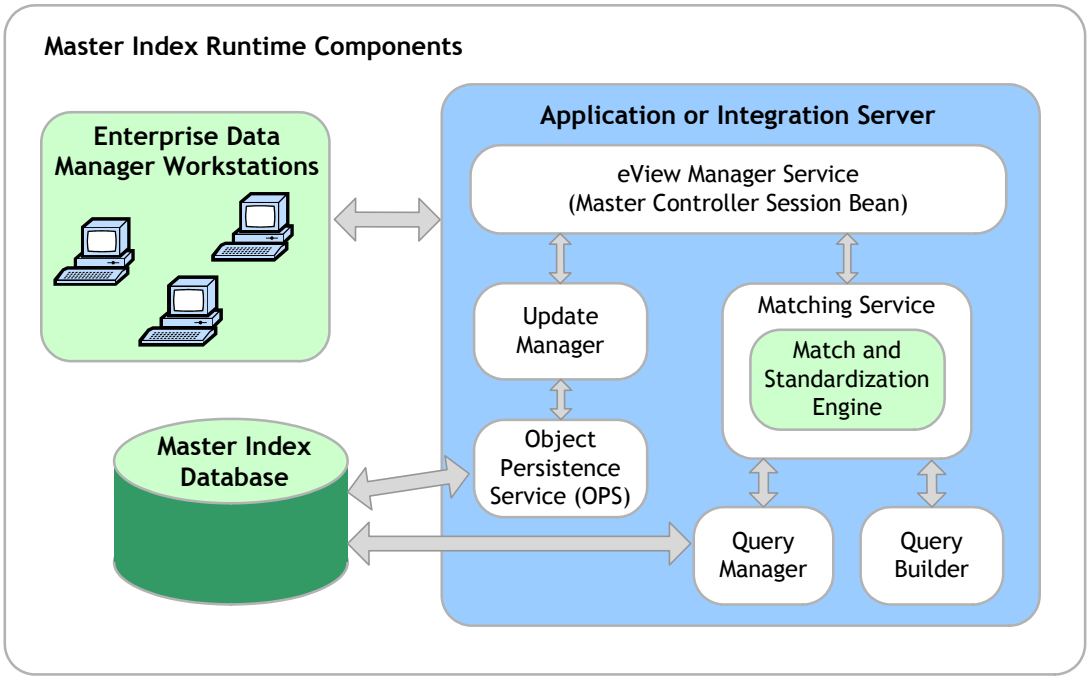
- **Matching Service** on page 30
- **eView Manager Service** on page 30
- **Query Builder** on page 30
- **Query Manager** on page 30
- **Update Manager** on page 30
- **Object Persistence Service (OPS)** on page 31
- **Database** on page 31
- **Enterprise Data Manager** on page 31

In addition, the master index uses the connectivity components defined in the eView Studio 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 Sun SeeBeyond Integration Server (IS) or the Sun Java System Application Server.

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

Figure 2 eView Studio Master Index Architecture



2.5.1 Matching Service

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

2.5.2 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 manages connectivity to the master index database. The configuration of the eView Studio 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 Studio Manager Service is defined in the Threshold file.

2.5.3 Query Builder

The Query Builder defines all queries available to the master index, including 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.

2.5.4 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.

2.5.5 Update Manager

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

2.5.6 Object Persistence Service (OPS)

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

2.5.7 Database

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.

2.5.8 Enterprise Data Manager

The Enterprise Data Manager (EDM) is a web-based interface that allows you to monitor and maintain the data in 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 Studio 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.

2.6 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 an external system. An enterprise record's SBR stores data from a combination of external systems and it represents the most reliable and current information contained in all system records for an enterprise record. An enterprise record consists of both system records and the SBR.

System Records

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

The Single Best Record

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

Objects in an Enterprise Record

In eView Studio, each system record and SBR in an enterprise record typically contains a set of objects that store different types of information about the business object. A record usually contains a parent object and several child objects. A record can have only one parent object, but can have multiple child objects and 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 primary 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 objects (for example, in address, phone, and alias objects). Each address would be stored in a different instance of an address object.

2.7 From eView Studio 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.

2.7.1 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 then create a new eView Studio application within that Project.
3. Define the object structure, operating environment, and certain runtime characteristics using the eView Wizard (**Chapter 4 "Creating the Master Index Framework"**).
4. If necessary, customize the configuration files (**Chapter 5 "Configuring the Master Index"**).

Note: *If you customize the Object Definition file, you might also need to make corresponding changes to the other configuration files. For example, if you add a new field to the Object Definition file 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. Define and build custom plug-ins, and specify the plug-ins in the appropriate configuration file (**Chapter 6 "Creating Custom Plug-ins"**).
6. Generate the master index (**Chapter 7 "Building the Application"**).
7. Create the database (**Chapter 8 "Creating the Database"**).
   - 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 "Defining Connectivity Components"**).
   - Create and define the components in the Connectivity Maps, such as Collaborations, Services, and External Applications.
   - Configure the Connectivity Maps.
9. Define the Environment (**Chapter 10 "Defining the Environment"**).
10. Create the Deployment Profile, deploy the Project, and define security (**Chapter 11 "Deploying the Project"**).

2.7.2 From XML to the Database

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

2.7.3 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 following.

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

2.7.4 From XML to the Connectivity Components

When you generate the eView Studio Project, several connectivity components are created, including a method OTD, Business Process methods, and an outbound OTD. All are 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 Business Process 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.

2.7.5 From XML to the Runtime Environment

The information you specify in the eView Studio configuration files is stored in the eGate Repository and is read at runtime when the domain is started. The only exception is the Object Definition file, which is stored only as a record of the object structure. You can modify the configuration files after moving to production. For the changes to take effect, you must regenerate the application and redeploy the Project to apply the changes to the server. 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 these files after moving to production might result in loss of data integrity or unexpected weighting and matching results.

2.8 Working with Project Components

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

Version Control

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

For more information about version control and checking files in and out, see chapter 3 of the *Sun SeeBeyond eGate Integrator User's Guide*.

Saving a Configuration File to the Repository

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

Note: *For more information about the XML Editor, see the* ***Sun SeeBeyond eView Studio Configuration Guide****.*

To save a file to the Repository

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

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

Validating XML Files

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

To validate XML syntax

1. After you save any changes to a configuration file to the Repository, keep the file open and right-click in the text of the file.

2 On the context menu that appears, click **Check XML**.

3 A message appears indicating the status of the validation and, if there were errors, includes a list of errors.

4 Fix any errors found in the file and revalidate.

To validate against the schema

1 After you save any changes to a configuration file to the Repository, right-click that file in the Project Explorer.

2 On the context menu that appears, click **Validate**.

3 A message appears indicating the status of the validation and, if there were errors, includes a list of errors.

4 Fix any errors found in the file and revalidate.

Copying, Cutting, and Pasting Files

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

Chapter 3

Installation

eView Studio is uploaded to the eGate Repository and is then installed in Enterprise Designer. This chapter provides instructions on installing eView Studio once the eGate environment is in place.

What's in This Chapter

3.1 Installation Overview

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

3.1.1 eView Studio Installation

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

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

3.1.2 Database Installation

All of the master index components you create using eView Studio 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 Studio tools, but it must be installed as a part of the master index implementation. For optimal performance, the database should be installed on its own server. The master index database can be installed on Oracle 9*i* or 10*g* and can run on any operating system platform supported by Oracle 9*i* or 10*g*.

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

To install the database, create a standard Oracle instance, using the sizing and distribution requirements obtained from the data analysis. After you run the eView Wizard and generate the Project, several SQL scripts are created. Running these scripts against the database creates the 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 Studio installation is a multi-stage process that includes the following:

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

3.3 Supported Operating Systems

This section lists the supported operating system requirements for each platform. The eView Studio **Readme.txt** file (located in the **eViewDocs.sar** file) contains the most up-to-date information for the supported platforms. eView Studio is available on the following operating systems.

- Sun Solaris 8, 9, and 10 with required patches (SPARC)
- Sun Solaris 10 (AMD Opteron)
- HP Tru64 V5.1A with patch 5 and V5.1B with required patches
- HP-UX 11.0 and 11i (11.11) on PA-RISC, and 11i v2.0 (11.23) on Itanium with required patches and parameter changes
- IBM AIX 5L, versions 5.2 and 5.3 with required maintenance level patches

- Microsoft Windows 2000 SP3 and SP4, Windows XP SP1a and SP2, and Windows Server 2003 SP1
- Red Hat Enterprise Linux AS 2.1 and AS 3 (Intel x86)
- Red Hat Enterprise Linux AS 3 and AS 4 (AMD Opteron)
- SUSE Linux Enterprise Server 8 and 9 (Intel x86)

3.4 System Requirements

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

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

3.4.1 Database Requirements

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 EDM require Internet Explorer 6.0 with SP1. If you will connect to the database using the Oracle eWay, you must install the Oracle eWay. To use the OCI driver with the Oracle eWay, you need to install the Oracle client on the Logical Host. Because the OCI driver supports both the most current Oracle database and any previous versions, Sun recommends that you use the most current version of the OCI driver. This assures compatibility between the various versions of any Oracle databases you have.

3.5 Software Requirements

eView Studio can be installed after you have done the following:

- Installed the Repository.
- Installed Enterprise Manager, including these files:
  A **eGate.sar**.
  B **FileeWay.sar**
  C **OracleeWay.sar**

D **eInsight.sar** (only if you want to install the eInsight sample Project for eView Studio or will use eView Studio functions in an eInsight Business Process)

3.5.1 Before Installing eView Studio

Before you install eView Studio, you must do the following:

1. Select the Window(s) computers that will host eView Studio. This must be a computer running Enterprise Designer, which only runs on Windows systems.
2. Determine the add-on applications, if any, you need.
3. Make sure the File eWay is installed. eView Studio relies on this eWay for the sample Project.
4. Determine whether you want to install the eView Studio client Project for eInsight BPM. If so, you must have eInsight BPM installed before installing eView Studio.
5. Determine whether to connect to the database using the application server or the Oracle eWay and whether to connect through a thin or thick client.
6. Make sure you have the appropriate administrator permissions to install eView Studio in Enterprise Designer and to update Enterprise Designer through the Update Center.
7. Before you begin the installation, exit all Windows applications.
8. If you are reinstalling eView Studio, back up any existing Projects by exporting them before beginning the installation. Once the installation is complete, import the existing Projects, regenerate any applications, and then rebuild and redeploy all related Projects.

3.6 Installing eView Studio

To install eView Studio, you must complete the following tasks:

- **Uploading eView Studio Components to the Repository** on page 40
- **Downloading eView Studio Components from the Repository** on page 44
- **Installing eView Studio in Enterprise Designer** on page 46

3.6.1 Uploading eView Studio Components to the Repository

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

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

To upload eView Studio components

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

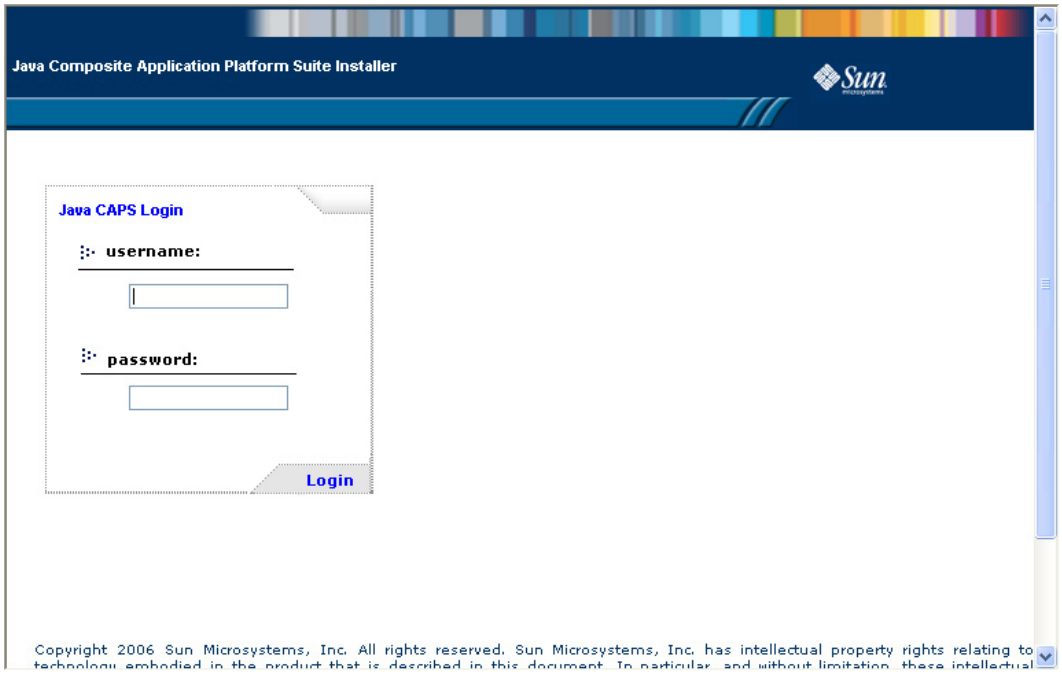
   <***hostname***> is the TCP/IP host name of the server where you installed the Repository—not the name of the Repository itself.

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

   When ready, press **Enter**.

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

Figure 3 Java CAPS Login Page



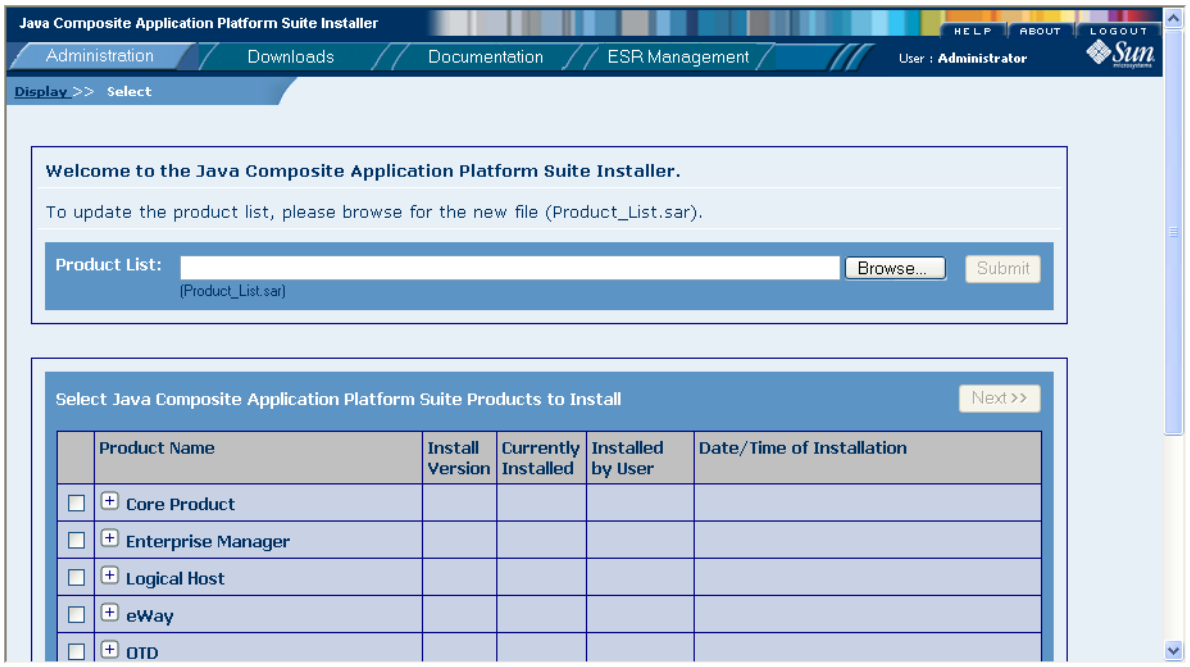
4. Enter your **username** and **password**.
5. When ready, click **Login**.

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

Figure 4 Administration Page

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

Figure 5 List of Products to Install



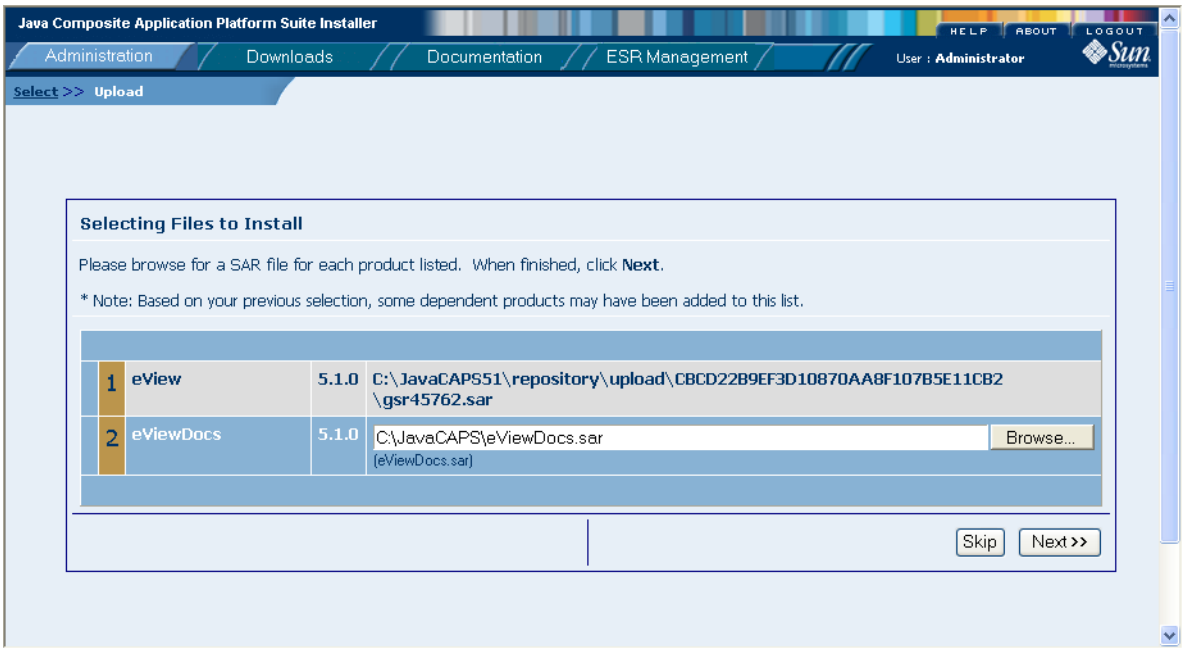
7 In the product list, expand Core Products, and then select the check box next to eView.

8 In the product list, expand Documentation, and then select the check box next to eViewDocs.

9 Click **Next** (in the bottom right section of the page).

The **Upload** page appears.

Figure 6 Selecting Files to Install



10 On the **Upload** page, browse for the requested file (either **eView StudioDocs.sar** or **eView Studio.sar**), and then click **Next**.

11 Repeat step 11 for the next requested file, and then click **Next**.

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

Figure 7 Installation Page

12 Do one of the following:

To download additional components of eView Studio using the installer, continue to the following set of procedures, **"Downloading eView Studio Components from the Repository" on page 44**.

To complete the eView Studio installation in Enterprise Designer, skip to **"Installing eView Studio in Enterprise Designer" on page 46**.

3.6.2 Downloading eView Studio Components from the Repository

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

Before performing any of these steps, make sure eView Studio is installed as described in **"Uploading eView Studio Components to the Repository" on page 40**.

Installing the eView Enterprise Manager Plug-in

The Enterprise Manager Monitor is a web-based interface you use to manage applications. The Monitor requires the plug-in specific to eView Studio to monitor and manage eView Studio applications. The plug-in enables the Monitor to target specific alert codes for eView Studio and is available from the Enterprise Manager.

To download the eView Enterprise Manager Plug-in

1. Start the Enterprise Manager.
2. From the toolbar of the Enterprise Manager Monitor, select **Configuration**.

   The User Preferences page appears.
3. Click the **Web Applications Manager** tab in the upper right section of the page.

   The Deploy New Management Application page appears.
4. Click the **Auto-Install from Repository** tab.
5. Enter the Repository logon information, and then click **Connect**.
6. In the **Management Applications available for installation from the Repository** list, select the check box next to eView Enterprise Manager Plug-in.
7. Click **Install**.

   In the Results list, verify that the eView Enterprise Manager Plug-in was installed properly.

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

Downloading eView Studio Reports

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

To download eView Studio reports

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

Accessing the eView Studio Documentation and Sample

When you uploaded the **eView StudioDocs.sar** file in previous steps, links to the documentation components of eView Studio were created on the **Documentation** tab of the installer. Perform any of the following steps to access the documentation files and sample.

- **To access the documentation files** on page 45
- **To download the Javadoc files** on page 46
- **To download the sample Project** on page 46

To access the documentation files

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

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

To download the Javadoc files

1. On the Suite Installer, click the **Documentation** tab.
2. Click **eView Studio**.
3. In the frame on the right side of the page, scroll to, and then click, **Download Javadoc**.
4. A dialog appears prompting you to open the file or save it to your computer. Extract the files to the directory where you want to store the files.
5. To access the documents, navigate to the directory where you extracted to files and then to **\eView Studio_Javadoc\html**.
6. Double-click **index.html**. This page provides links to all Javadoc pages.

To download the sample Project

1. On the Suite Installer, click the **Documentation** tab.
2. Click **eView Studio**.
3. In the frame on the right side of the page, scroll to, and then click, **Download Sample**.
4. A dialog appears prompting you to open the file or save it to your computer. Choose **Save**, and then choose a location for the file set.
5. To import and implement the sample Project, see **Chapter 13 "Implementing the eView Studio Sample"**.

3.6.3 Installing eView Studio in Enterprise Designer

The final step in installing eView Studio is updating Enterprise Designer with the eView Studio files. Enterprise Designer must be installed on the machine on which you are installing eView Studio. This step is performed using the Update Center, which is a

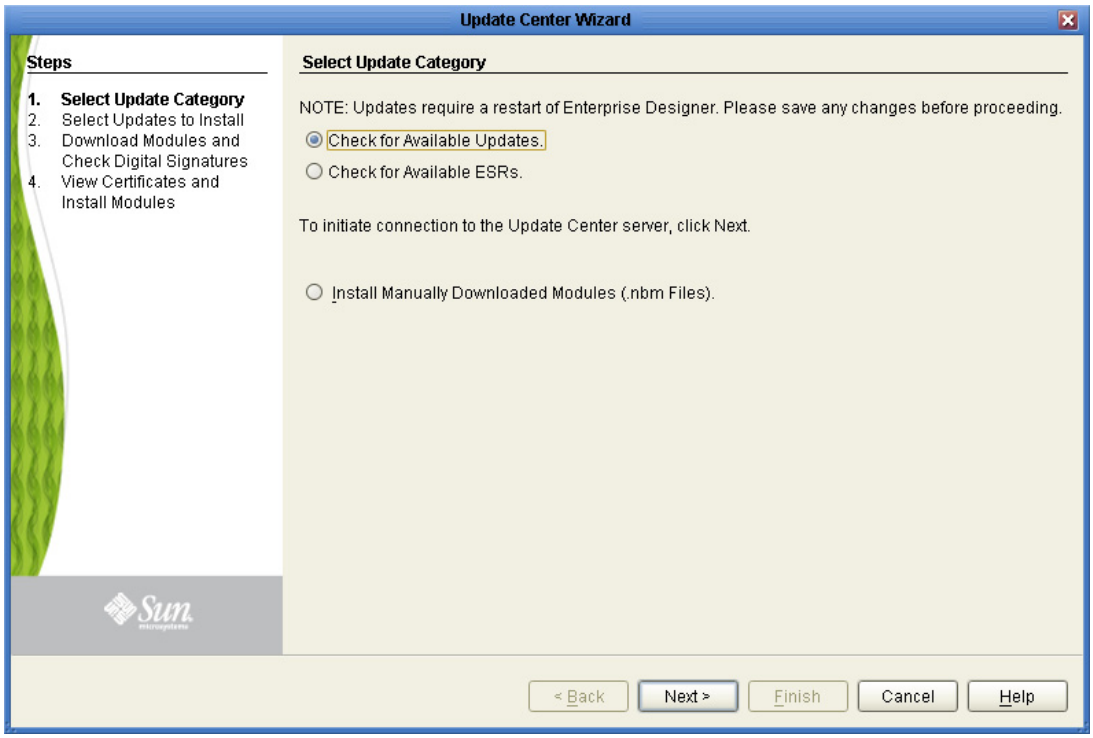
tool in Enterprise Designer that allows you to install add-on modules into Enterprise Designer.

To install eView Studio in Enterprise Designer

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

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

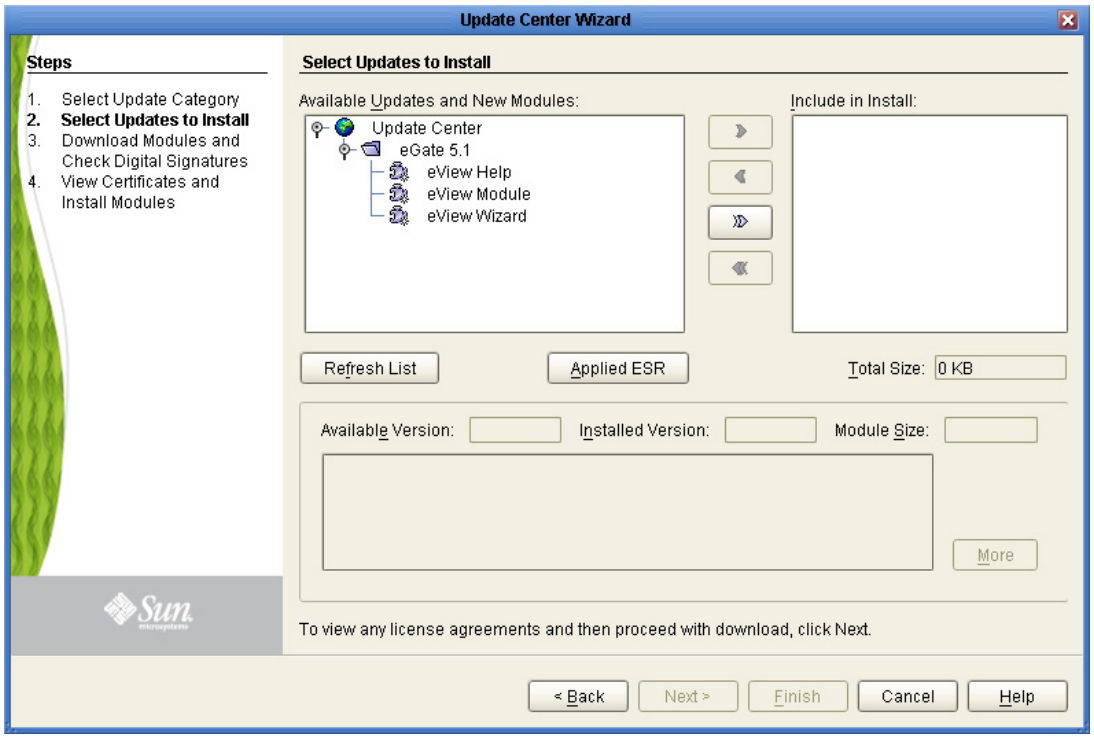
Figure 8 Update Center wizard - Select Update Category



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

   The **Select Modules to Install** window appears (see Figure 9).

Figure 9 Update Center Wizard - Select Modules to Install



4 Do one of the following:

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

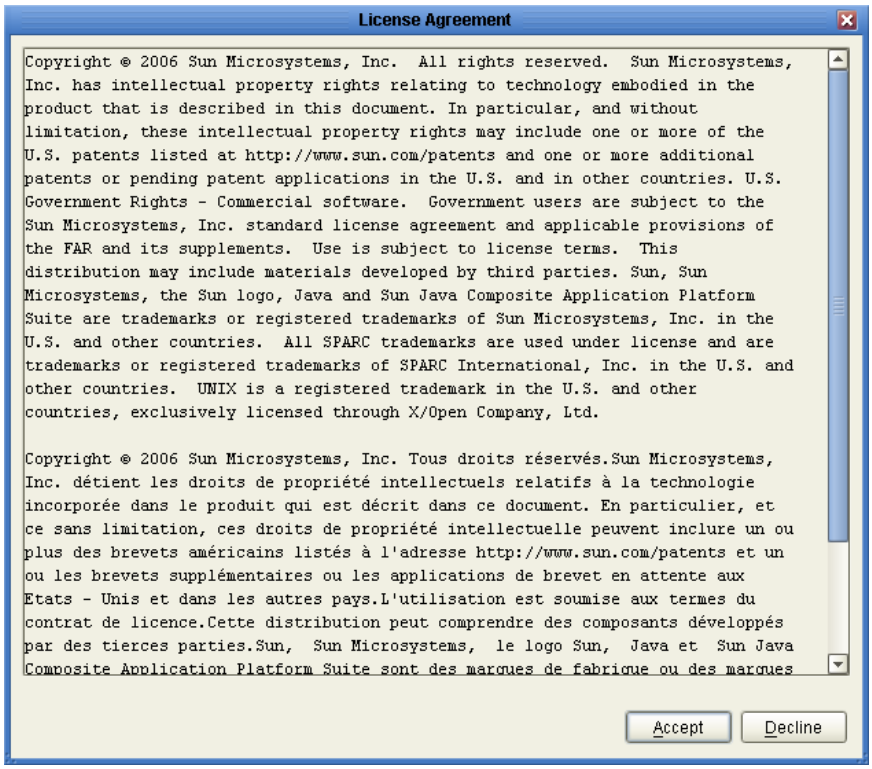
Note: *To select each eView Studio module, hold down the* ***Control*** *key and click each module.*

The eView Studio files move to the **Include in Install** list.

5 Click **Next**.

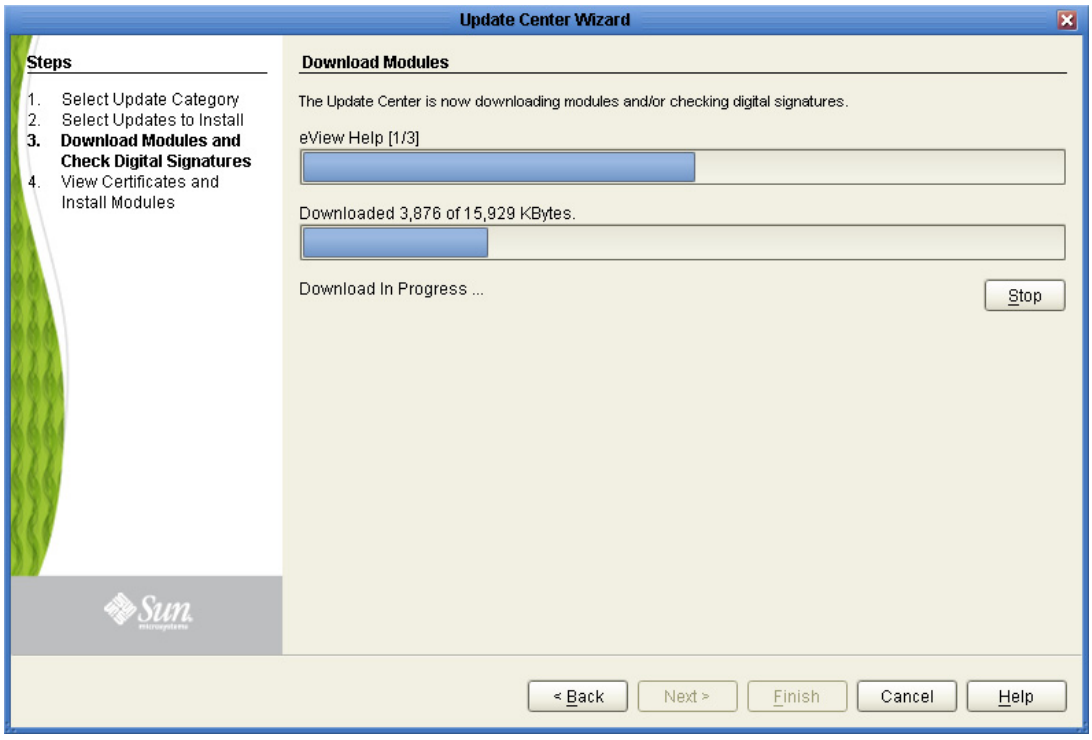
The License Agreement window appears (see Figure 10).

Figure 10 Update Center Wizard - License Agreement



6 On the **License Agreement** window, click **Accept**. The **Download Modules** page appears (see Figure 11).

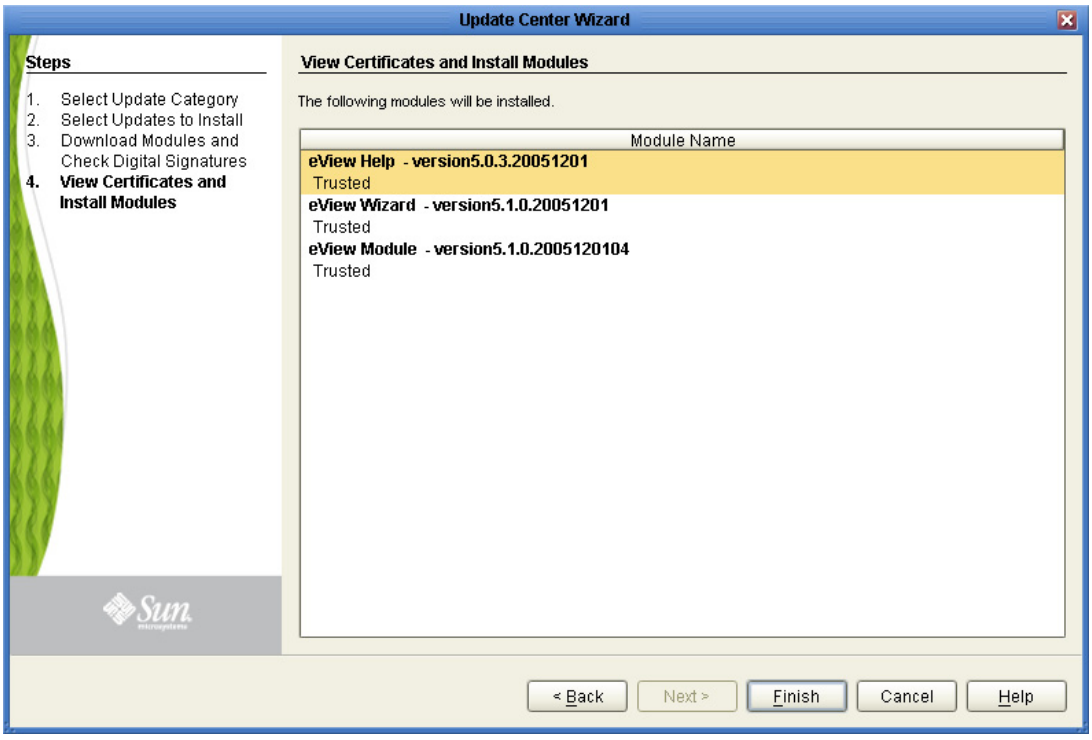
Figure 11 Update Center Wizard - Download Modules



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

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

Figure 12 Update Center Wizard - View Certificates and Install Modules

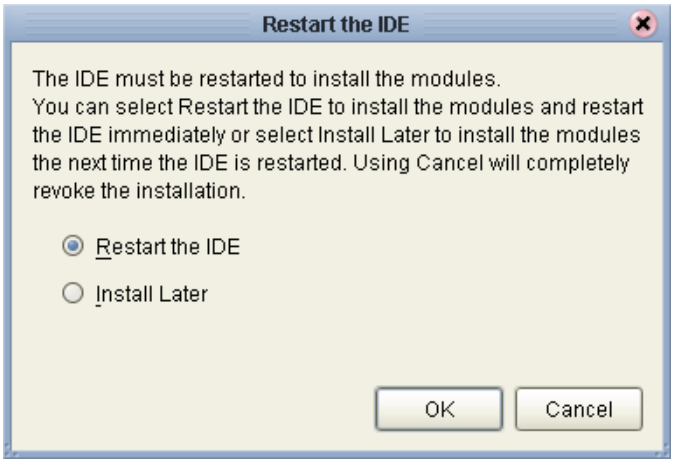


8 Click **Finish**.

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

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

Figure 13 Update Center Wizard - Restart the IDE



10 When Enterprise Designer is restarted, the sample Projects appear in the Project Explorer.

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

3.7 Final Steps

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

For the Oracle eWay

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

Important: *A domain must be created for eView Studio before performing these steps.*

To provide access to the Oracle thick client for IS implementations

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

For the Sun Java System Application Server

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

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

Chapter 4

Creating the Master Index Framework

The first step in creating a master index uses 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.

What's in This Chapter

4.1 The eView Studio Project

The eView Wizard runs in the context of a Project in Enterprise Designer. 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 Studio Project.

- Object Definition
- Configuration Files
- Database Scripts (for processing codes and system information)
- Application .jar files

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 Studio system. You can also specify characteristics about the appearance of the Enterprise Data Manager (EDM).

When you complete the wizard, eView Studio 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 Enterprise Designer. **Chapter 5** provides an overview of these files. The *Sun SeeBeyond 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 Studio editors in 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 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.

Important: *You should be familiar with Enterprise Designer and eGate Projects before creating the master index.*

4.2.2 eView Wizard Toolbar Buttons

The toolbar buttons described in Table 2 provide 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.

Table 2 eView Wizard Toolbar Buttons

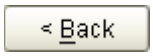
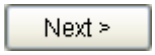
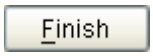
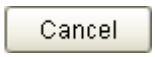
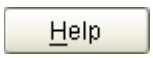
Button	Command	Function
	Add Field	Add a new field under the selected object.
	Delete	Deletes the selected object or field. If you delete an object, all fields in that object are also deleted.
	Templates	Opens a template menu, from which you can select a Company or a Person template with predefined fields and sub-objects.

4.2.3 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	Returns to the previous step in the wizard. This button is disabled on the first step.
	Next	Goes to the next step in the wizard. This button is disabled on the last step.
	Finish	Saves configuration information, created Project files, and closes the wizard. This button is only enabled on the last step.
	Cancel	Closes the wizard without saving the configuration information.
	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 Studio Project. For additional information about analyzing, planning, and designing eGate components, see the *Sun SeeBeyond eGate Integrator Deployment Guide*.

4.3.1 Data Analysis

Before running the eView Wizard, 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 external system that needs to share data with the eView Studio master index. At a minimum, each extracted data record should include all fields used for matching. Sun or a qualified third party performs the preliminary data analysis. It is important that Sun receive 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 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 Studio Project in Enterprise Designer, you must analyze the business requirements of the project and determine which Project components 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.

An additional consideration is whether to integrate the eView Studio methods into an eInsight Business Process or into eVision web pages.

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)
- The fields that are required to enter data or that are required for queries
- The fields that will appear on reports

- The fields that must be unique to the primary object
- The fields that will be used for matching
- Any special formatting requirements, such as character types, the data type, minimum and maximum values, and field size
- The fields that will appear on EDM search and search results windows
- The processing codes for the source systems being integrated into 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 configuration files for the master index. To create the initial master index configuration, follow these steps.

4.4.1 Step 1: Create a Project

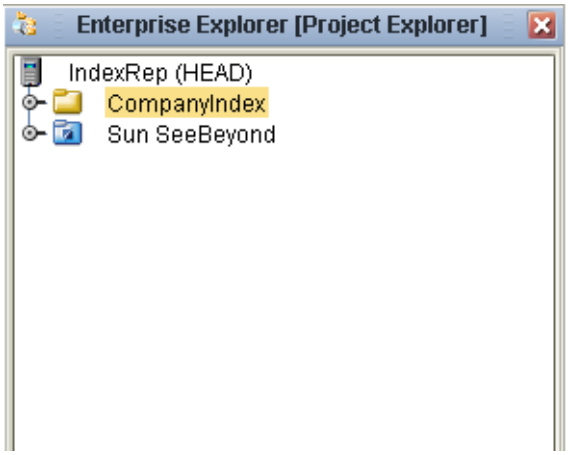
Before you can access the eView Wizard, you must create and name a new Project in Enterprise Designer.

To create a Project

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



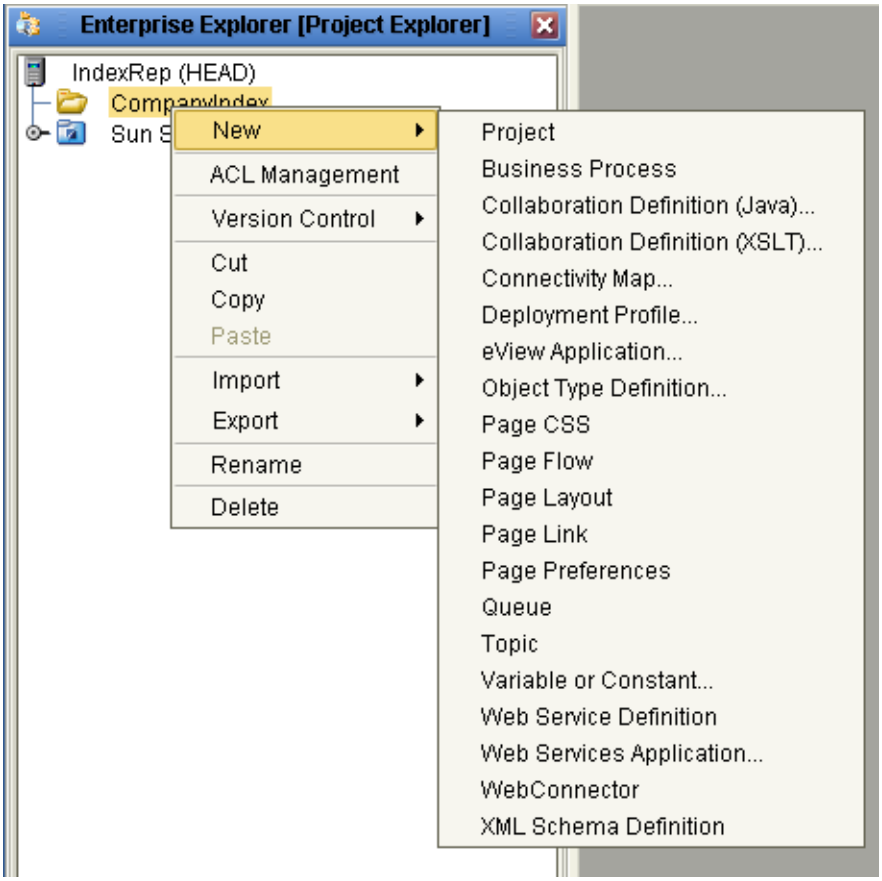
4.4.2 Step 2: Launch the eView Wizard

Once you create a Project for the master index, you can launch the eView Wizard and begin defining the eView Studio 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 Enterprise Designer 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 Studio Application"**.

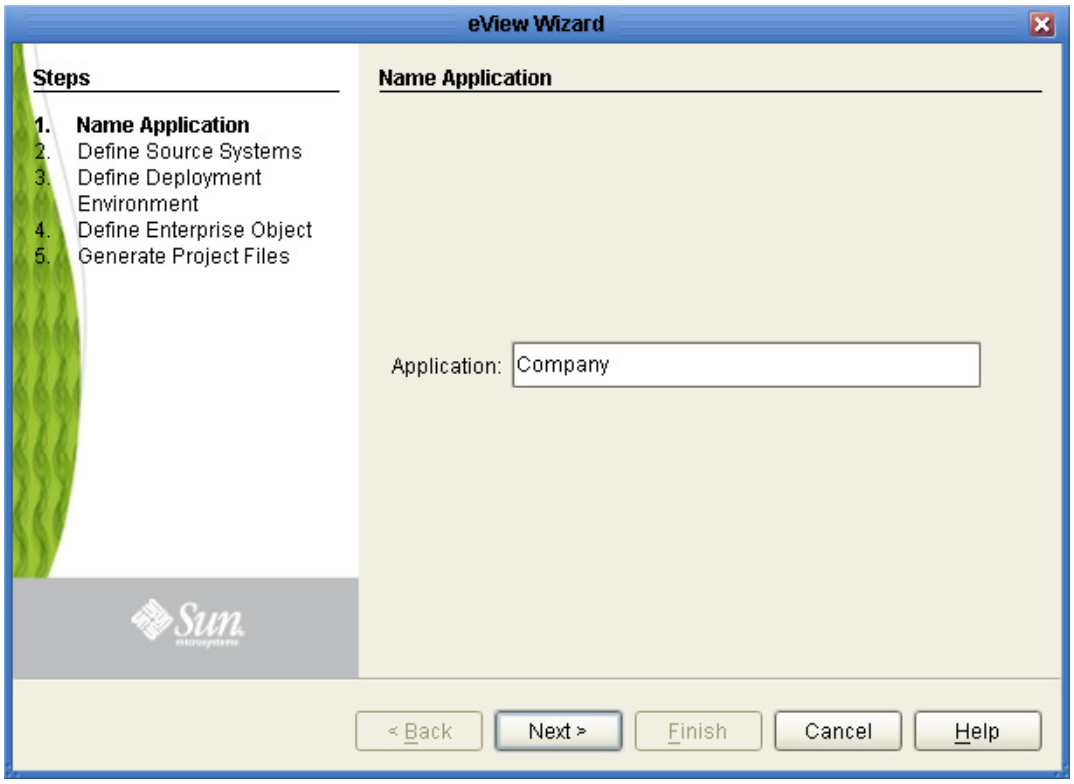
4.4.3 Step 3: Name the eView Studio Application

Each master index you create is an eView Studio 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 and its corresponding database table. The name must follow Oracle naming requirements for database tables. Do not use characters restricted by Oracle, Java, or XML.

To name the eView Studio application

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

Figure 16 eView Wizard - Name Application



2 In the **Application** field of the **Name Application** window, enter a name for the new eView Studio 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 Studio 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 processing codes for the source systems that will be integrated into the new master index system.

To define source systems

1 Complete **"Step 3: Name the eView Studio 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 Studio system, and then click **Add**. You can enter any of the following characters:

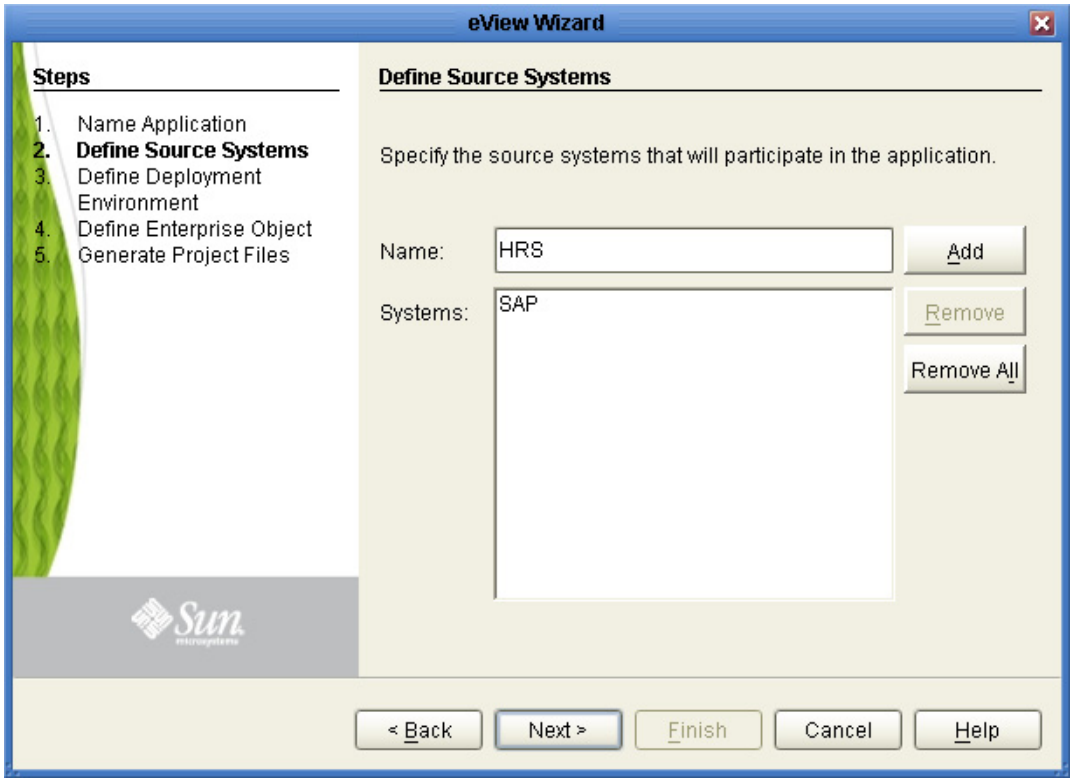
- ! _ ~ ( ) { } + ` # $ % & : ; - /
- a-z

- A-Z
- 0-9
- þ ÿ Þ ß 'à-ö ø-ý À-Ö Ø-Ý

The value you entered appears in the **Systems** box. **Figure 17 on page 61** illustrates system codes being added in the eView Wizard.

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



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"**.

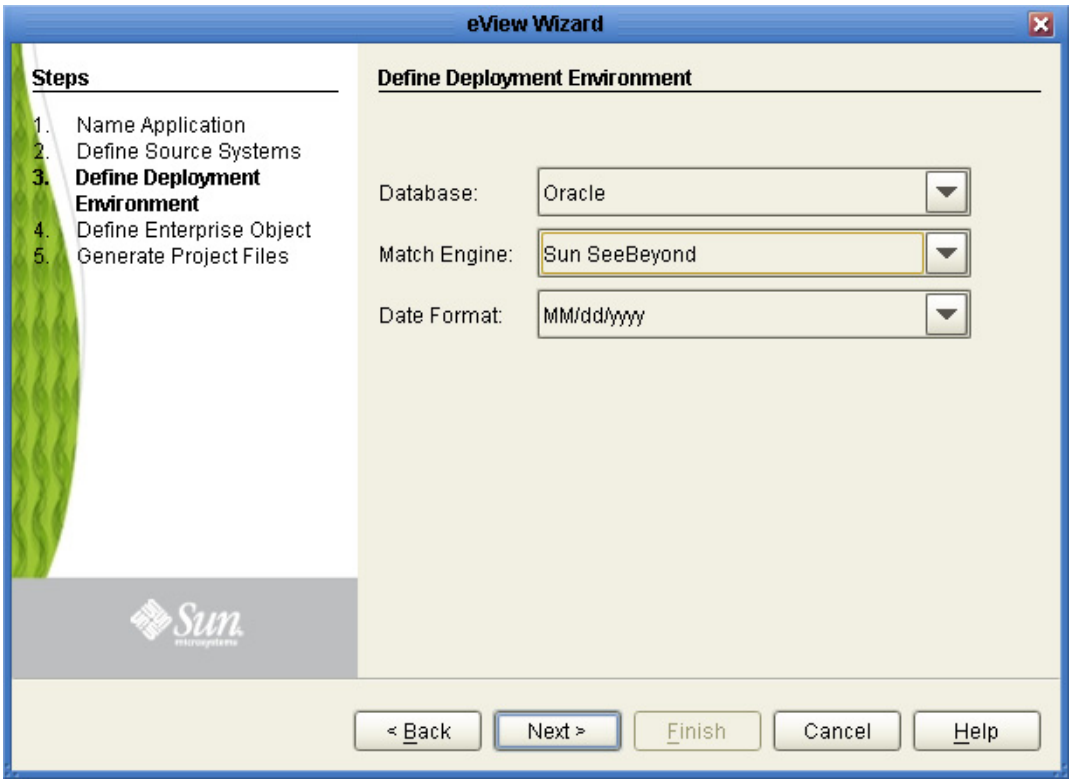
4.4.5 Step 5: Define the Deployment Environment

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



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. Currently, the only option is the Sun SeeBeyond Match Engine.

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. Child objects are not required if all information is stored under the parent object.

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 63
- **Creating Objects from a Template** on page 65
- **Deleting an Object from the Structure** on page 67

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

Important: *The names of database constraints are created based on the parent and child object names. Due to Oracle naming restrictions, the length of the parent object name plus the length of any of the child object names must be 21 characters or less.*

Creating Undefined Objects

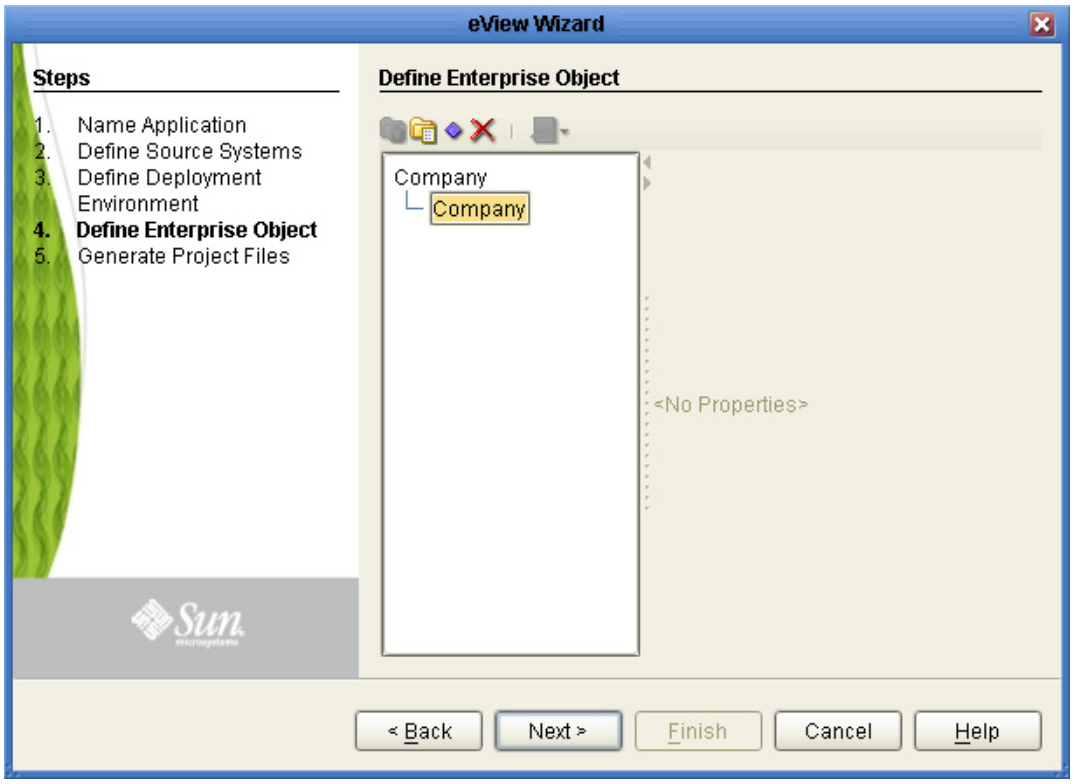
When you create undefined objects, you create an empty object with no predefined fields or child objects.

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

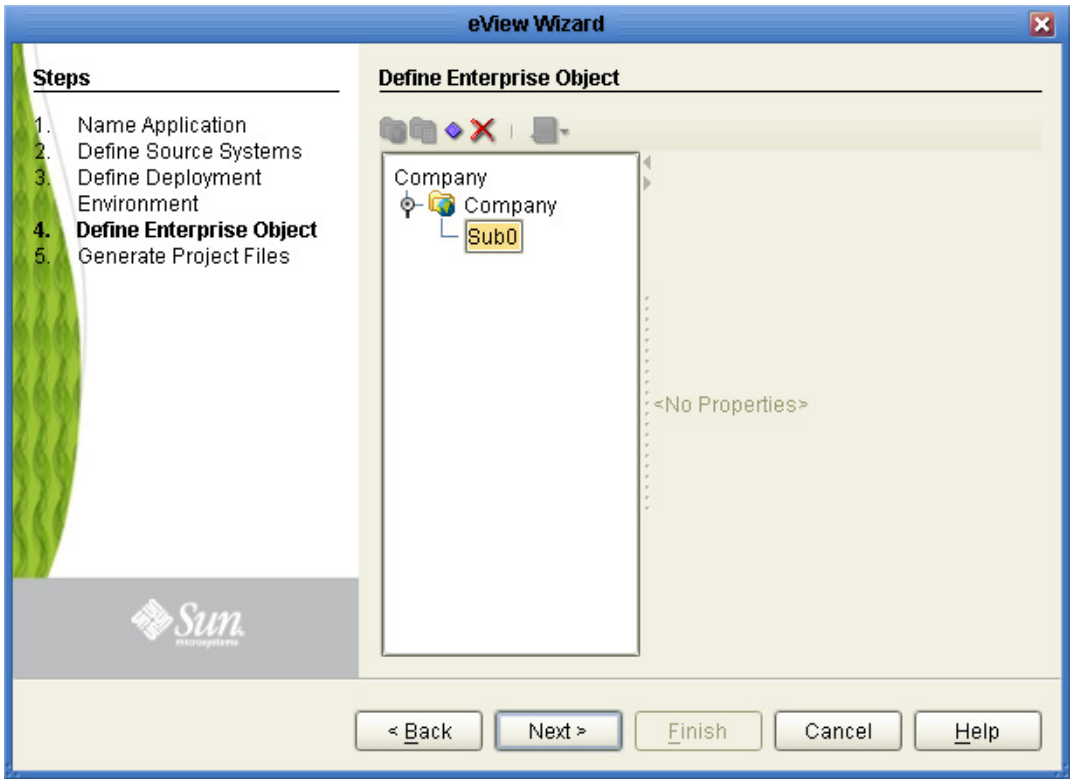


2 Accept the default name by pressing **Enter**.

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

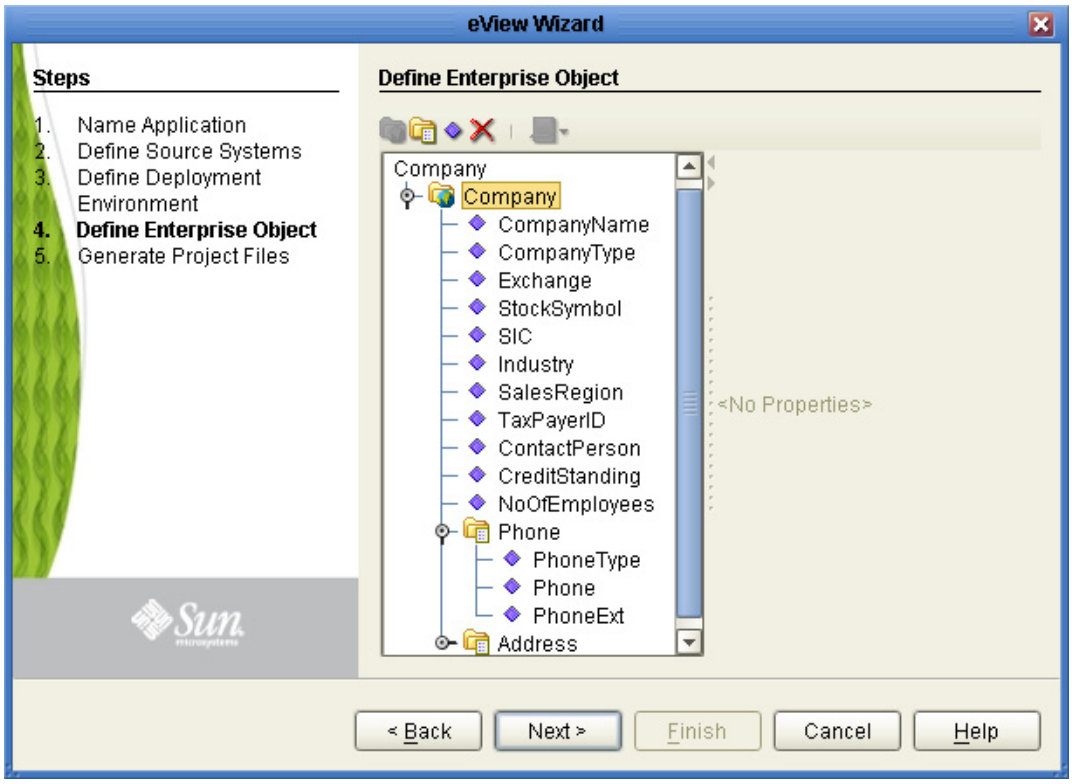
When you create objects from a template, secondary objects and fields are predefined. You can modify the objects, fields, and field properties in a template to suit your processing needs.

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.

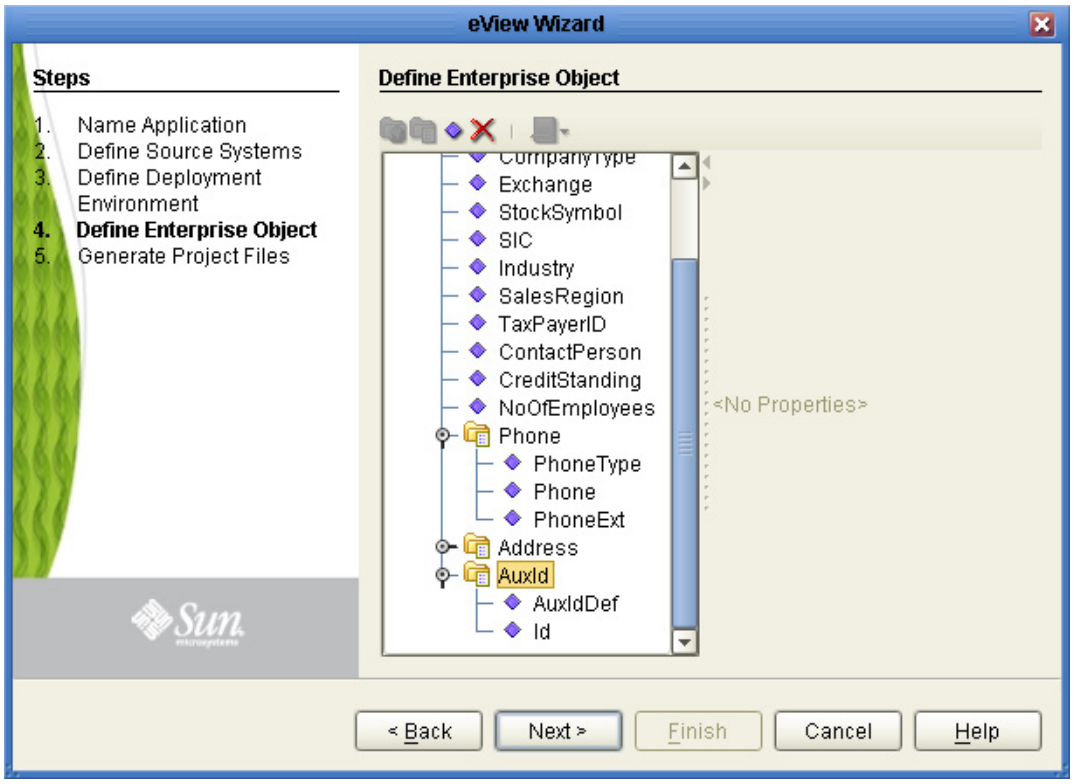
Figure 21 eView Wizard - Company Template



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 in 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 define the parent and child objects, you can perform any of the following actions to define the fields for those objects.

- **Adding a Field** on page 68
- **Configuring Field Properties** on page 70
- **Deleting a Field** on page 75

Adding a Field

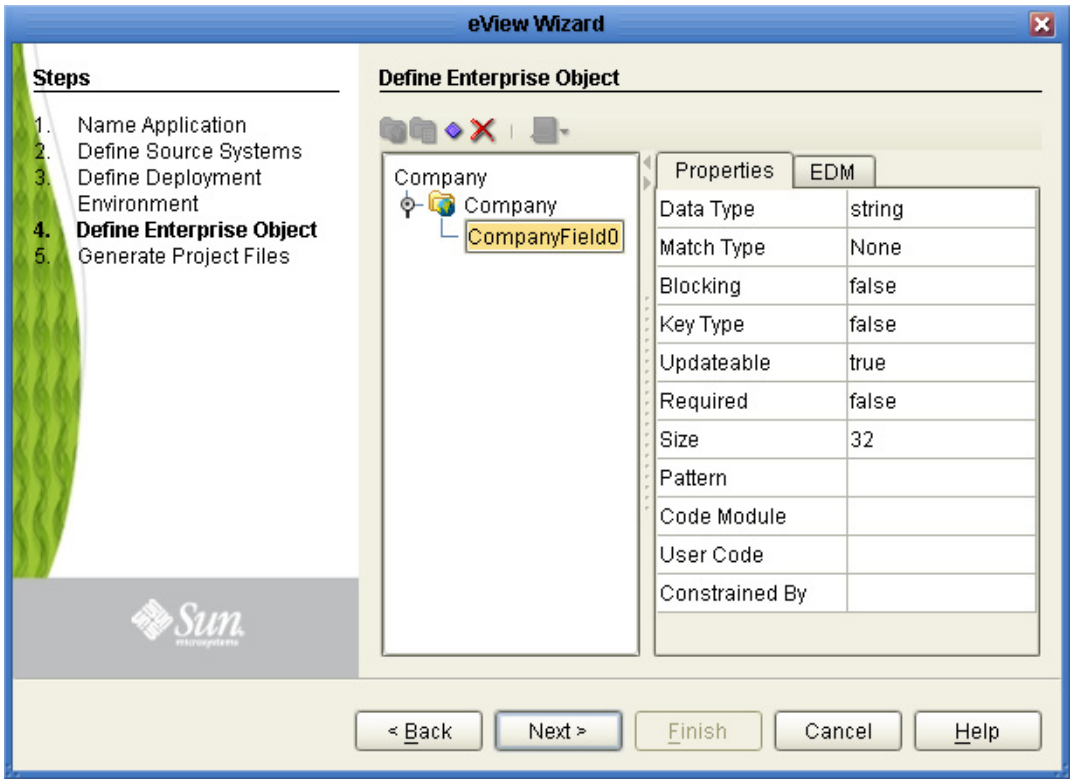
If you created an empty object in **"Step 6: Define Parent and Child Objects"**, you must create each field that belongs to the object. If you created objects using a predefined template, you can add new fields to the object if needed.

To add 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



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

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

Important Field Name Restrictions:

- *eView Studio automatically creates a field for each object named* ***<object>Id****, where* ***<object>*** *is the name of an object or sub-object. You cannot create fields with those names. For example, you cannot create a field named "AddressId" if there is an Address object in the object structure.*
- *If you enter a field name longer than 20 characters, a warning dialog appears. While Oracle can handle names up to 30 characters, eView Studio 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 C "eView Wizard Match Types"**.
- *Do not use characters or names restricted by Oracle, Java, or XML in field names.*

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.

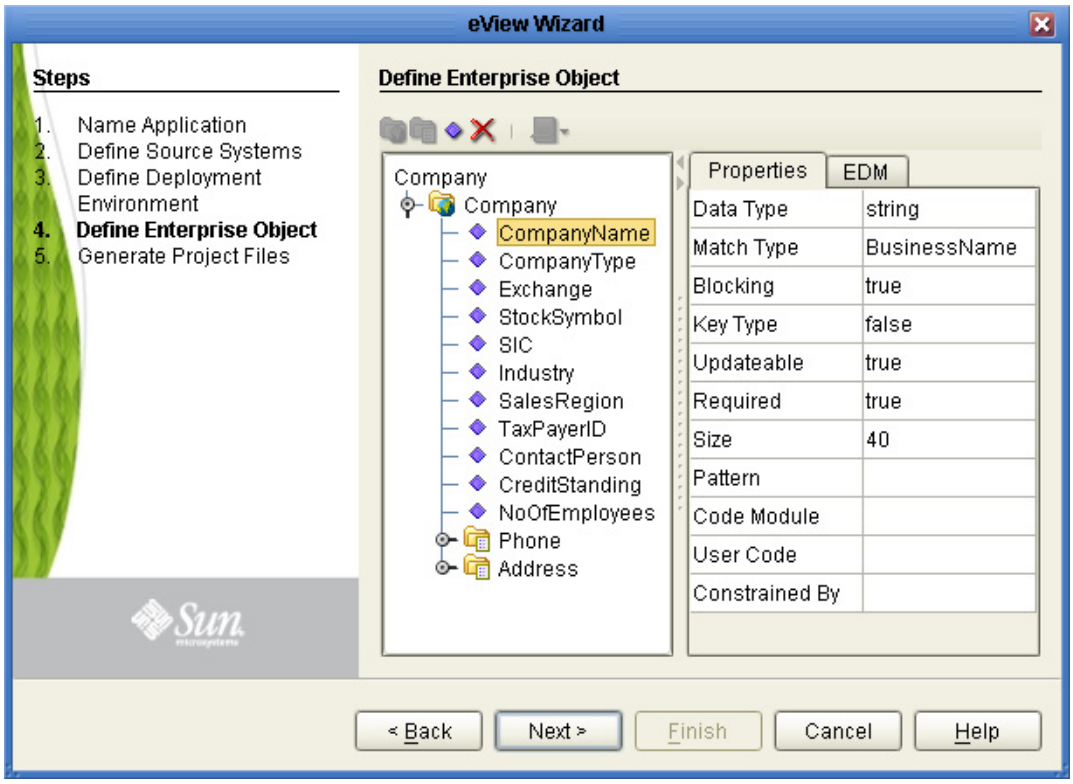
To configure field properties

1. Complete the steps under **"Adding 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 71**.

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

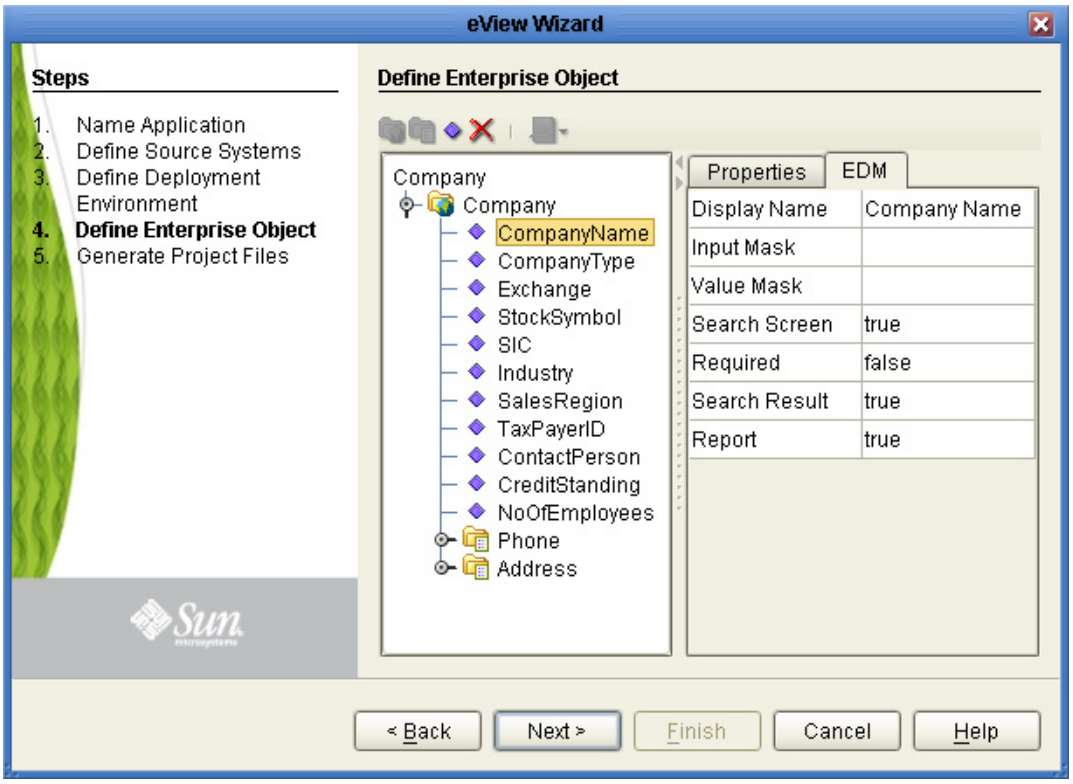
Figure 24 Field Properties Page



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

   To see a list of descriptions and restrictions for each property, see **Table 6 on page 73**.

Figure 25 Field EDM Page



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 Studio data type of the field. The following data types are supported: ▪ **string** - Fields of this type contain a string of characters. ▪ **date** - Fields of this type contain a date value. ▪ **float** - Fields of this type contain a floating point integer. ▪ **int** - Fields of this type contain an integer. ▪ **char** - Fields of this type contain a single character. ▪ **boolean** - Fields of this type can contain either "true" or "false".

Table 5 Field Properties

Property	Description
Match Type	The type of matching to be performed against the field, if the field is to be used for match weight generation. You must define at least one field for matching or no weights will be generated. ***Note:*** *The match types you specify here define the structure of the Match Field file, including the match string. The match types in the Match Field file might differ from the eView Wizard match types. See* **eView Wizard Match Types** on page 194 *for information about the available options for this field and how the wizard match types correlate to the Match Field file types.*
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.
Key Type	An indicator of whether the field is used to identify unique objects. For example, a business index might store several addresses for each business. Each address is assigned an address type and each business can only have one address of each type. Specify **true** if the field is a unique record identifier, or **false** if it is not. Key type fields should also be required fields (see below), unless a combination of fields are specified as key types for an object. ***Note:*** *It is recommended that each child object contain a key type field, but this is not required. If child objects do not contain one or more key type fields, each enterprise object might accumulate a very large number of child objects depending on the survivor strategy used.*
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, that field should 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.

Table 5 Field Properties

Property	Description
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 the J2SE Platform. You might want to define patterns for date, telephone, or SSN fields. Note that for the EDM, the pattern is further restricted by the value entered for the input mask described in Table 6. If no input mask is specified, all regex patterns are supported.
Code Module	The identification code for the drop-down list that appear for this field in the EDM. ***Note:*** *This must match an entry in the* ***code*** *column of the sbyn_common_header database table and, by default, an entry for the code you enter is created in the* ***Code List*** *database script. You can further customize code lists in the script 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.*
Constrained By	The name of the field that contains the corresponding **User Code** value (described above). **User Code** and **Constrained By** are used in conjunction to define a drop-down list for the field that has the **User Code** value and to validate the field that has the **Constrained By** value against definitions for the field with the **User Code** value. For example, if you store non-unique IDs such as credit card numbers or insurance policy numbers, you could create a field named **ID Type** that has a **User Code** value matching a code in the sbyn_user_code table. This gives the **ID Type** field a drop-down list based on the definitions in the sbyn_user_code table. You could then create a field named **ID** that would be constrained by the **ID Type** field. Any IDs you enter would be validated against the value of the **ID Type** field.

Table 6 EDM Properties

Property	Description
Display Name	The name of the field as it will appear on the EDM.

Table 6 EDM Properties

Property	Description
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. Use the value mask (described below) to strip the punctuation from the value before storing it in the database. 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. Note that the value you enter can further restrict the data pattern for the field (this is the Pattern field property described in Table 5). The following character types can be used. ▪ **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 (see above). 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, specify a value mask of DDDxDDxDDDD to strip the dashes before storing the SSN. A value mask is not required for date fields.
Search Screen	An indicator of whether the field appears on the search windows of the EDM. Specify **true** to display the field or **false** to hide it.
Required	An indicator of whether the field must be populated on the search windows of the EDM when performing a search. This field can only be modified if the Search Screen property is set to **true** (see above). Specify **true** to make the field required or specify **false** to make it optional. You can also specify **oneof** to create a group of fields of which at least one must be populated to perform a search. (Be sure to specify **oneof** for each field in the group.) ***Tip:*** *If a field is required for a search, the field should also be required for creating a record (by specifying true for the* ***Required*** *property on the Properties page). Otherwise, searches performed from the EDM could result in no matches even though possible matches exist.*

Table 6 EDM Properties

Property	Description
Search Result	An indicator of whether the field appears on the search results windows of the EDM. Specify **true** to display the field, or **false** to hide it.
Report	An indicator of whether the field appears on the reports generated from the EDM. Specify **true** to display the field on the reports; otherwise specify **false**.

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.

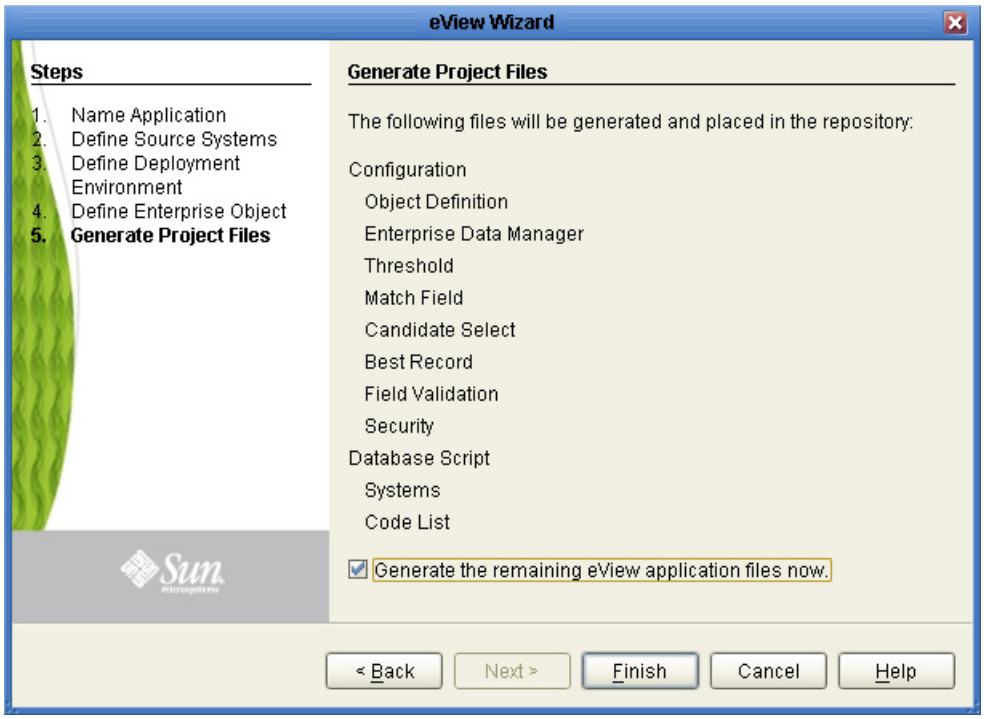
4.4.8 Step 8: Generate the Project Files

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 additional Project files at this time (such as the .jar files and OTDs) or to wait until you have customized the configuration files for the eView Studio 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



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 Studio configuration files are placed in the Project for the master index. Verify that the configuration files are customized correctly for your implementation. If you need to modify the configuration files, **Chapter 5** provides an overview of these files. See the *Sun SeeBeyond eView Studio Configuration Guide* for information about modifying the files. You can also create custom plug-ins to further customize how your eView Studio application processes and stores data.

Chapter 5

Configuring the Master Index

The eView Wizard creates several nodes in the eView Studio Project that represent configuration files for the master index. Before generating the eView Studio 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 *Sun SeeBeyond eView Studio Configuration Guide*.

What's in This Chapter

5.1 Configurable Options

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

5.1.1 Object Definition

By customizing the Object Definition file, you can configure the structure of the data stored in the master index. This file contains the configuration of all objects in the

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

5.1.2 Enterprise Data Manager

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

5.1.3 Query Definitions

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

5.1.4 Standardization and Matching Rules

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

5.1.5 Survivor Calculator

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

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

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

5.1.6 Update Policies

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

5.1.7 Field Validations

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

5.1.8 Enterprise-wide Unique Identifiers

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

5.2 About the eView Studio Configuration Files

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

The configuration files include the following:

- **Object Definition File** on page 80
- **Candidate Select File** on page 80
- **Match Field File** on page 80
- **Threshold File** on page 80
- **Best Record File** on page 81
- **Field Validation File** on page 81
- **Security File** on page 81
- **Enterprise Data Manager File** on page 81

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

5.2.1 Object Definition File

In the eView Wizard, you define the objects and fields contained in the object structure, along with properties of those fields. The information you specify is written to the Object Definition file in the eView Studio 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 properties of each field, such as length, data type, whether it is required, whether it is a unique key, and so on. This file contains one parent object; all other objects must be child objects to that parent object. The object structure defined in the Object Definition file determines the structure of the database tables that will store object data, the structure of the Java API, and the structure of the OTD generated for the Project.

5.2.2 Candidate Select File

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

5.2.3 Match Field File

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

5.2.4 Threshold File

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

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

5.2.5 Best Record File

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

5.2.6 Field Validation File

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

5.2.7 Security File

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

5.2.8 Enterprise Data Manager File

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

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

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

5.3 Modifying the eView Studio Configuration Files

You might need to modify the configuration files after you review them. Make sure that when you modify the configuration files, you use the **Check Out** and **Check In** commands to maintain version control. If you open and modify a file without first checking the file out, a warning appears when you try to save the file. This warning lets you save and check out the file in one step. Also, be sure to verify that the modifications are valid by verifying the XML syntax and using Enterprise Designer's validation function to validate the file against the schema. After modifying each file, save the changes to the Repository.

There are a few restraints on modifying these files. In addition to the general rules listed below, the match engine you choose might place other requirements on customizations. Be sure to review *Implementing the Sun SeeBeyond Match Engine with 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 the eView Studio application and redeploy the eView Studio Project. You must also refresh any Collaborations in client Projects that reference the eView Studio server Project.

5.4 Match Engine Configuration Files

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

Chapter 6

Creating Custom Plug-ins

eView Studio provides the ability to create custom update procedures to be performed on a record during and after standard eView Studio processing. These procedures are defined as Java classes in the Custom Plug-ins module.

This chapter describes how to create custom procedures using this module and how to configure eView Studio to use the custom procedures.

What's in This Chapter

6.1 Custom Processing

You can add custom processing to the master index using the Custom Plug-ins module of an eView Studio Project. Plug-ins can be used to customize field validations, update policies, match processing logic, and record retrieval, as well as create custom components for the master index, such as a custom phonetic encoders, block pickers, or query builders.

The following sections describe custom plug-ins that define custom processing. These are explained more fully in the *Sun SeeBeyond eView Studio Configuration Guide*.

6.1.1 Update Policies

For the primary transactions performed by the master index you can define additional custom processing to perform against the record that results from a transaction. The policies you define are invoked by the Update Manager and are applied to the resulting

records after they are processed by the survivor calculator. The modifications made to a record by an update policy determine how the record is stored in the database. Using the Custom Plug-ins function, you can create additional Java classes to support the update policies you define.

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

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

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

Enterprise Merge Policy

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

Enterprise Unmerge Policy

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

Enterprise Update Policy

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

Enterprise Create Policy

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

System Merge Policy

The system merge policy defines additional processing to perform after two system objects are merged. The processing defined in this file acts against the surviving enterprise record of the merge (and not the system record). In the **SystemMergePolicy** element of the Best Record file, enter the fully qualified name of this custom plug-in.

System Unmerge Policy

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

Undo Assumed Match Policy

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

6.1.2 Field Validations

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

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

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

6.1.3 Field Masking

There might be instances where you want to mask certain data in records from general users of the Enterprise Data Manager and only allow access by administrators. To do this, you can create a custom plug-in that displays asterisks (or other symbol) in place of the fields values on the EDM. Once you define the custom plug-in, specify the name of the custom plug-in Java class in the object-sensitive-plug-in-class element of the Enterprise Data Manager configuration file (see chapter 9 of the *Sun SeeBeyond eView Studio Configuration Guide*).

6.1.4 Match Processing Logic

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

Execute Match Methods

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

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

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

Custom Logic Methods

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

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

Custom Logic Plug-in Requirements

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

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

Threshold File Customization

If you create a custom plug-in that defines custom processing logic for the execute match methods, you must specify those custom plug-ins in the Threshold file of the eView Studio Project. If you create a plug-in for customizing logic in the execute match methods used by Collaborations or Business Processes, specify the name of that class in the logic-class element. If you create a plug-in for the EDM, specify the name of that class in the logic-class-gui element. For example:

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

For more information about the Threshold file, see the *Sun SeeBeyond eView Studio Configuration Guide*.

6.2 Custom Components

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

6.2.1 Survivor Calculator

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

6.2.2 Query Builder

The query builder defines the different types of queries that can be used in the master index. You can implement custom queries using custom plug-ins. To create a new query builder, you must define a class that extends the base abstract **com.stc.eindex.querybuilder.QueryBuilder** and then specify that class in a **query-**

builder element in the Candidate Select file. The exception thrown is **com.stc.eindex.querybuilder.QueryBuilderException**. The following methods must be implemented. For more information about query-related Java classes, see the Javadocs provided with eView Studio.

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

6.2.3 Block Picker

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

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

6.2.4 Pass Controller

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

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

6.2.5 Match Engine

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

- Implement the **com.stc.eindex.matching.MatcherAPI** interface to communicate with the match engine.

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

6.2.6 Standardization Engine

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

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

6.2.7 Phonetic Encoders

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

6.3 Custom Plug-in Exception Processing

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

For more information about these exception classes, see the Javadoc provided with eView Studio.

6.4 Implementing Custom Plug-ins

eView Studio provides a simple method of incorporating custom Java code into an eView Studio 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 Studio 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 Studio Project, click the **Custom Plug-ins** folder and then right-click.
2. Select **New** from the context menu that appears.
3. Enter the name of the custom plug-in and then click **OK**.

   The custom plug-in file appears in the Java Source Editor with the first line already entered ("**package com.stc.eindex.user;**").
4. Create the custom processing rules using Java code.
5. Close and save the file.
6. Repeat these steps for each plug-in you need to create.
7. Build the custom plug-ins, as described under **"Building Custom Plug-ins"**.
8. Specify the name of the custom plug-in in the appropriate eView Studio configuration file.

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

Building Custom Plug-ins

In order for the custom plug-ins you create to become a part of the eView Studio application, you must build the plug-ins. This compiles the Java code and incorporates it into the application files. Compiling errors for custom plug-ins are not written to a log. An error message trace appears on a console window alerting you to the errors that occurred.

To build custom plug-ins

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

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

Chapter 7

Building the Application

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

What's in This Chapter

7.1 Generated Application Components

Generating the master index 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 Studio uses the Object Definition to generate these components based on the configuration you defined. The generated components include the following.

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

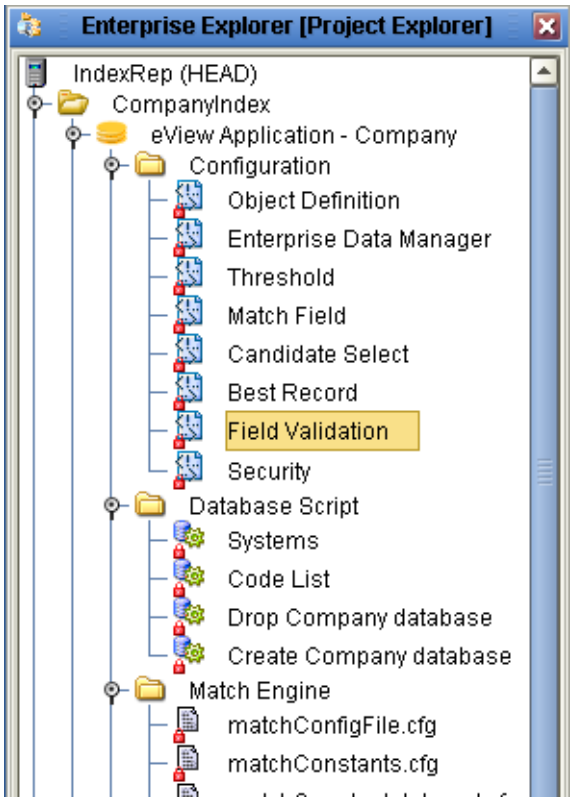
7.2 Generating the Application

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

To generate the application

1. Save all configuration changes to the Repository.
2. Right-click the eView Studio application in the **Project Explorer** pane to display the application context menu (shown in Figure 27).

Figure 27 Generate Context Menu



3. Select **Generate**. eView Studio creates the new Project components. This might take a few minutes.

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

4. Save the new components to the Repository.

5 If there are any client Projects with Collaborations that reference the eView Studio server Project, refresh those Collaborations.

6 If there were any changes to how incoming data is processed (such as changes to the output OTD), re-import the .jar files.

A Open the Collaboration in the Collaboration Editor (Java) and click **Import JAR Files**. The Add/Remove Jar Files dialog appears.

B For each eView Studio .jar file, highlight the filename on the Add/Remove Jar Files dialog, and then click **Remove**.

C For each file to re-import, click **Add**, double-click the eView Studio server Project, select the .jar file, and then click **Import**.

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

7 If you are using command line reports, do the following:

A Export the regenerated **<Application>_stc_eindex_client.jar** and **<Application>_stc_eindex_util.jar** files to the **lib** subdirectory in the reports home directory.

B In the **lib** subdirectory, rename the file **<Application>stc_eindex_client.jar**. to **stc_eindex_client.jar** and rename the file **<Application>stc_eindex_util.jar**. to **stc_eindex_util.jar**.

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

Chapter 8

Creating the Database

eView Studio 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.

What's in This Chapter

8.1 Database Scripts

The eView Wizard creates two SQL scripts based on information you specified about code lists and 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 Script** node of the eView Studio Project, and are named **Systems**, **Code List**, **Create <*application_name*> database**, and **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 database distribution. You can also create new database scripts if needed.

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.

8.2.1 Database Platform Requirements

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

8.2.2 Operating System Requirements

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

8.2.3 Hardware Requirements

This section describes the minimum recommended hardware configuration for a database installation is one of the following options. These requirements are based on the minimum requirements recommended by Oracle for the installation of a Typical installation. Depending on the size of the database and expected volume, you should increase these recommendations as needed.

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

Important: *Disk space recommendations do not take into account the volume and processing requirements or the number of users. These are minimal requirements to install a*

generic database. At a minimum, the empty database and the database software will require 2.5 GB of disk space.

8.3 Database Structure

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

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

For a complete description of the database tables, see Chapter 3, "The Database Structure", in the *Sun SeeBeyond 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. The eView Studio database should be created in its own tablespaces. The following sections describe some of the analyses to perform along with things to consider when designing the database.

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 guides include information about monitoring and fine-tuning your database, including tuning memory, swap space, I/O, CPU usage, block and file size, and so on.

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

8.4.3 Common Table Data

Common table data analysis involves gathering information about the abbreviations used for specific data elements in each sending system, such as system codes and codes for certain attributes about the objects in your database. For example, if you are indexing person objects, there might 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

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

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 should install these tables into different tablespaces, depending on the original size and expected volume of the database.

Database Sizing

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

Database Distribution

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

Database Indexes

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

8.5 Creating the Database

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

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 and User

Before beginning this step, be sure that the correct version of Oracle is installed on the database server (version 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 define tablespaces, including their sizes and locations; extents; and dump file, log file, and rollback file sizes and locations. Make sure these issues have been thoroughly analyzed and designed before creating the database.

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

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

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

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

8.5.3 Step 3: Customize the Database Scripts

The database script that installs the database components specific to eView Studio 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. Sun 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 Script** 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 Studio Project.

To define a system

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

The file opens in the text editor.

2 For each "insert" statement, modify the values clause according to the column descriptions in Table 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. This field accepts any of the following characters: ▪ ! _ ~ ( ) { } + ` # $ % & : ; - / ▪ a-z ▪ A-Z ▪ 0-9 ▪ þ ÿ Þ ß 'à-ö ø-ý À-Ö Ø-Ý
description	A brief description of the system, or the system name. ***Tip:*** *It is best to keep this short if possible; these values appear in the tree views on the Enterprise Data Manager and can cause the box containing the tree views to increase in width to accommodate all characters.*
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 default maximum length of the LID database columns is 25. If the system generates longer local IDs, be sure to increase the length of all LID columns in the database.*
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 the J2SE Platform. Note that the pattern specified here might be further restricted by the input mask described below.
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. If you use an input mask, you should also define a value mask (see below) to remove the punctuation from the stored value.
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 is 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 Script** 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 Script** node and then double-click the **Code List** file.
2. The file opens in the text editor.
3. In the **Code List** file, scroll to the following line.

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

   The statements following this line must be customized.
4. 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'),
   ```
5. Create the entries for the module using the following syntax:

   ```
   tCode('V', 'code', 'code description'),
   ```

   where “code” is the processing code of the data element and “code description” is 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 and 4 for each code list type defined in the file.

   If you specified additional code list fields in the Object Definition and Enterprise Data Manager files, 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 Script** 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).

6 Save and close the 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 element. This is the value that appears in the drop-down list.
format	The required data pattern for the field that is constrained by the user code. For more information about possible values and using Java patterns, see “Patterns” in the class list for **java.util.regex** in the Javadocs provided with the J2SE Platform. Note that the pattern might be further restricted by the value of the input mask described below.

Table 8 SBYN_USER_CODE Table Description

Column Name	Description
input-mask	A mask used by the EDM to add punctuation to the constrained field. For example, the input mask **DD-DDD-DDD** inserts a hyphen after the second and fifth characters in an 8-digit ID. If you use an input mask, you should also use a value mask to strip the punctuation for database storage (see below). The following character types can be used. ▪ **D**—Numeric character ▪ **L**—Alphabetic character ▪ **A**—Alphanumeric character
value-mask	A mask used to strip any extra characters that were added by the input mask for database storage. The value mask is the same as the input mask, but with an "x" in place of each punctuation mark. Using the input mask described above, the value mask is **DDxDDDxDDD**. This strips the hyphens before storing the ID.

Creating a Custom Script

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

To create a custom script

1. In the eView Studio Project, right-click the **Database Script** node.
2. In the Database context menu, select **New**.
3. Enter the name of the script, and then click **OK**.

   The new script appears under the **Database Script** node.
4. Double-click the new script.

   The text editor appears.
5. In the text editor, create the SQL script to insert the custom data.
6. Close the file and select **Yes** to save the data.

8.5.4 Step 4: Modify the Database

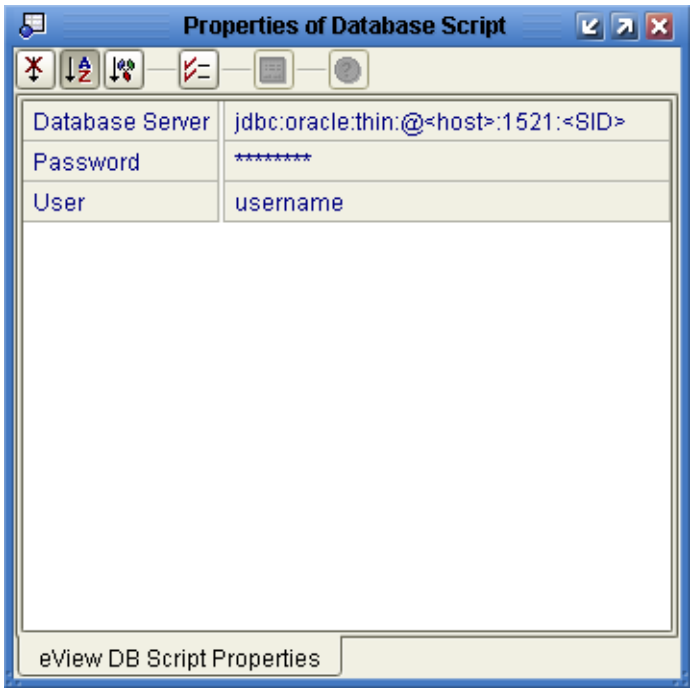
After you create the database instance and customize the database scripts, you can create the master index tables and insert the custom data.

To modify the database

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

   The **Properties of Database Script** dialog appears as shown in Figure 28.

Figure 28 Database Properties Dialog



2 In the **Properties of Database Script** dialog, enter the following information:

- In the **Database Server** field, change ***<host>*** to the database server name and change ***<SID>*** to the SID name of the database you created in **"Step 2: Create an Oracle Database and User"**. You can use "localhost" as the database server name if the database resides on the same machine as the Enterprise Designer.
- In the **Password** field, enter the password of the administrator user you created when you created the database (creating an administrator user is described under **"Step 2: Create an Oracle Database and User" on page 98**).
- In the **User** field, enter the administrator user's logon ID.

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

3 Close the dialog by clicking the "X" icon in the upper right corner of the dialog.

4 Right-click **Create *<app_name>* Database** (where ***<app_name>*** is the name of the eView Studio 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 "0", with padded zeroes added to the left to make the EUID number the correct length (for more information, see "Threshold Configuration" in the *Sun SeeBeyond 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 Tables

If you need to remove the database tables created in **Step 4: Modify the Database** on page 104, you can run the "drop script" created by the eView Wizard. This is useful while testing the master index implementation.

To delete tables

1. Right-click **Drop <*app_name*> database** (where **<*app_name*>** is the name of the eView Studio application).
2. Select **Run**.
3. On the confirmation dialog, click **OK**.
4. If the database is running on the Oracle 10g platform, open a SQL prompt and run the following command.

```
PURGE RECYCLEBIN;
```

Chapter 9

Defining Connectivity Components

Once the eView Studio server 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, Business Processes or Collaborations, and external systems. This chapter describes the connectivity components used in conjunction with an eView Studio master index and how to configure those components using the Enterprise Designer tools.

What's in This Chapter

9.1 Connectivity Overview

The Project that defines the eView Studio application is known as the *server* Project; the Projects that define external system connectivity with eView Studio through Collaborations or Business Processes are known as *client* Projects.

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

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

9.1.1 Connectivity Components

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

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

eView Studio Server Project Connectivity Components

The eView Studio server Project can include the following connectivity components.

- **Connectivity Map** - Graphically describes the relationship between the web application components and the master index application components.
- **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 Studio 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 Studio Project.
- **Oracle eWay** - Provides connectivity to the eView Studio database. This component is required if eView Studio is running on the Sun SeeBeyond Integration Server. It is optional if eView Studio is running on the Sun Java System Application Server, which allows you to define the database connection pool through the server.
- **JMS Topic** - A message destination conforming to the *publish-and-subscribe* (pub/sub) messaging paradigm. In this case, eView Studio publishes to the topic to broadcast messages to external systems. This component is optional.

Client Project Connectivity Components

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

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

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

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

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

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 Studio Project and for any client Projects that reference the eView Studio application. This section describes how to define connectivity components for eView Studio server and client Projects, including Collaborations, Business Processes, Services, JMS Topics, and so on. Perform the following tasks to configure connectivity for the master index and connected systems.

- **Defining eView Studio Application Connectivity Components** on page 110
- **Defining Collaboration Client Connectivity Components** on page 113
- **Defining eInsight Client Connectivity Components** on page 122

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

9.2.1 Defining eView Studio Application Connectivity Components

In the eView Studio server 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 Studio Project, add components to the map, and then connect those components. Perform the following tasks to create and configure the eView Studio Project Connectivity Map.

- **Creating the eView Studio Server Connectivity Map** on page 110
- **Connecting eView Studio Server Connectivity Map Components** on page 111

Creating the eView Studio Server Connectivity Map

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

To create the eView Studio server Connectivity Map

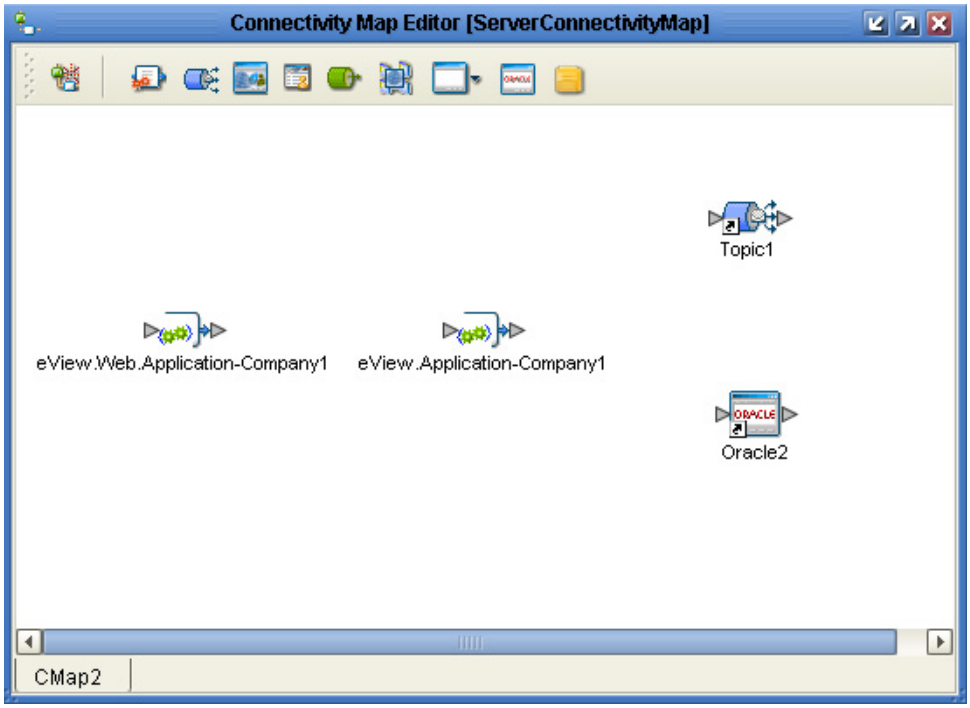
1. In Enterprise Explorer, select the eView Studio server Project to which you want to add the Connectivity Map.
2. Right-click to display the **Project** context menu.
3. Select **New > Connectivity Map** to add a **Connectivity Map** icon to the Project and display the Connectivity Map Editor window.
4. Change the name of the Connectivity Map.
   A. Click the **Connectivity Map** icon in the Project Explorer.
   B. Change the default name to the name you want to use.
   C. On the confirmation message, click **OK**.
5. Drag the **eView.Web.Application-<application_name>** icon from the Project Explorer onto the Connectivity Map Editor canvas.
6. Drag the **eView.Application-<application_name>** icon from the Project Explorer onto the canvas to the right of the web application icon.
7. (Optional) Add an Oracle eWay for database connectivity by doing the following:
   A. On the Connectivity Map Editor toolbar, click the **External System** icon.
   B. From the drop-down list, select the check box for **Oracle External Application**. An Oracle External Application icon appears on the toolbar.
   C. Drag the new icon from the toolbar onto the canvas to the right of the **eView.Application-<application_name>** icon.

Important: *If eView Studio is running on the Sun SeeBeyond Integration Server (IS), you must add an Oracle eWay to configure the database connection pool. It is optional with the Sun Java System Application Server.*

8. (Optional) Add a Topic to publish messages from the master index to external systems, select the **Topic** icon in the Connectivity Map Editor toolbar, and then drag it onto the canvas to the right of the **eView.Application-<application_name>** icon.

The Connectivity Map should now look similar to Figure 29.

Figure 29 Server Connectivity Map



9 Save the Connectivity Map to the Repository.

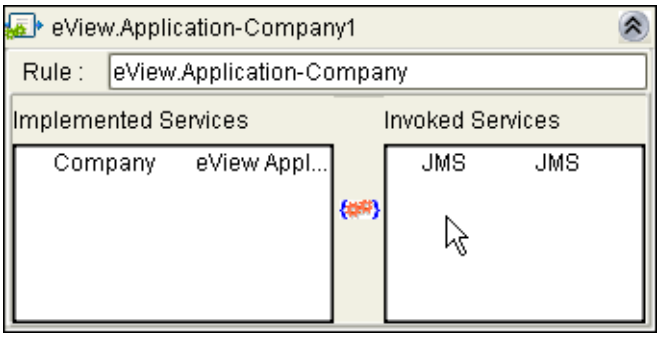
Connecting eView Studio Server Connectivity Map Components

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 complete all of the steps in **"Creating the eView Studio Server Connectivity Map" on page 110** and have the eView Studio Connectivity Map visible on the Connectivity Map Editor.

To connect eView Studio server Connectivity Map components

1 In the Connectivity Map Editor, double-click the **eView.Application-<application_name>** icon. The Service Binding dialog appears.

Figure 30 Collaboration Client Service Binding Dialog



2 Place the cursor over the arrow to the right of the **eView.Web.Application-<application_name>** icon in the Connectivity Map until the cursor turns into a hand.

3 Click the arrow and drag it to the connector next to Implemented Services on the Service Binding dialog.

4 If the Connectivity Map contains and Oracle eWay, configure the eWay.

   A Click and drag the arrow from the right side of Invoked Services on the Service Binding dialog to the **Oracle External Application** icon in the Connectivity Map.

   B Double-click the Oracle eWay icon on the new connector bar.

   C On the eWay Connections dialog, click **OK**.

   D On the Oracle eWay Properties window, enter the properties or accept the default values (for more information, see the *Sun SeeBeyond eWay Adapter for Oracle User's Guide*).

   E Click **OK**.

5 If the Connectivity Map contains a JMS Topic, configure the Topic.

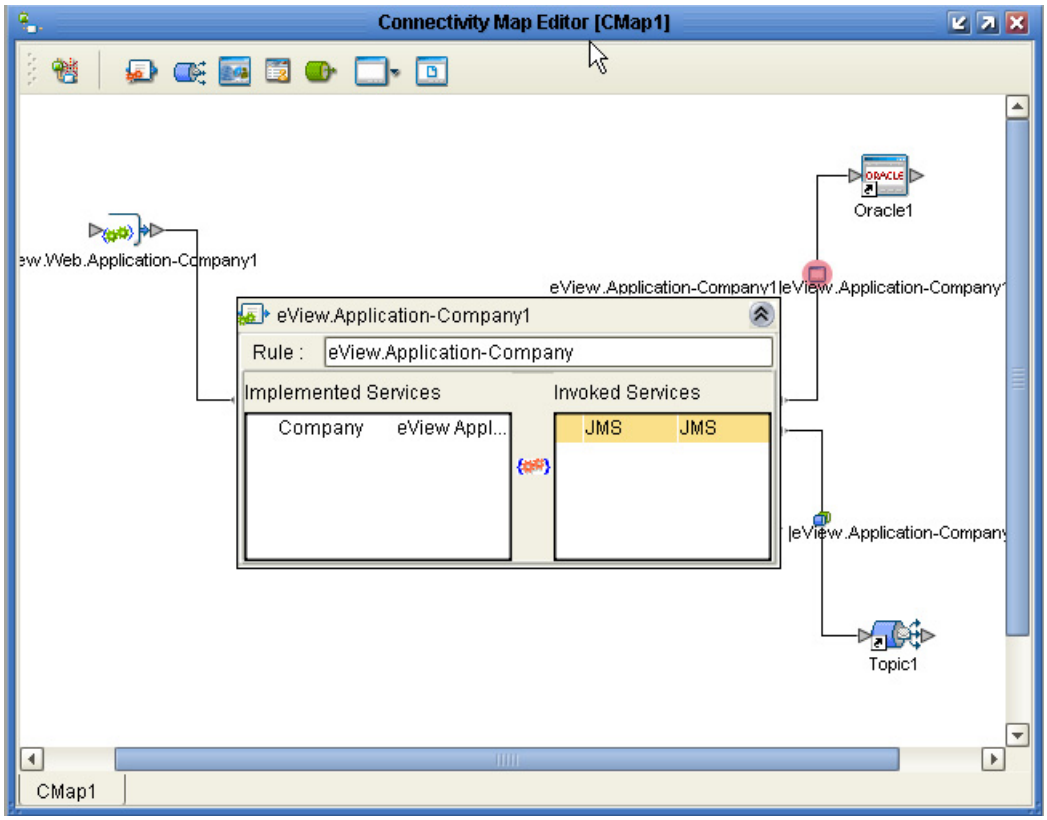
   A Click and drag **JMS** from the Invoked Services box to the **Topic** icon in the Connectivity Map.

   B Double-click the JMS Client Connection icon on the new connector bar.

   C Configure the JMS Client Connection (for more information, see the *Sun SeeBeyond eGate Integrator JMS Reference Guide*).

Figure 31 illustrates the completed Connectivity Map for the eView Studio Project.

Figure 31 Server Connectivity Map with Connections



6 Close the Service Binding dialog, and then save the Connectivity Map to the Repository.

9.2.2 Defining Collaboration Client Connectivity Components

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

- **Adding eView Studio Methods to a Java Collaboration** on page 114
- **Creating the Collaboration Client Project Connectivity Map** on page 115
- **Connecting Connectivity Map Components** on page 117
- **Incorporating the JMS Topic into the Connectivity Map** on page 119
- **Configuring the Outbound Collaboration** on page 121

Adding eView Studio Methods to a Java Collaboration

This section describes how to incorporate the eView Studio method OTD into Java Collaborations for external systems. For a complete reference of the methods included in the eView Studio OTD, see the *Sun SeeBeyond eView Studio Reference Guide*. You can use additional eView Studio methods, which are described in the Javadocs provided with eView Studio.

Before beginning this procedure, create the OTD for the incoming messages.

To add eView Studio methods to a Java Collaboration

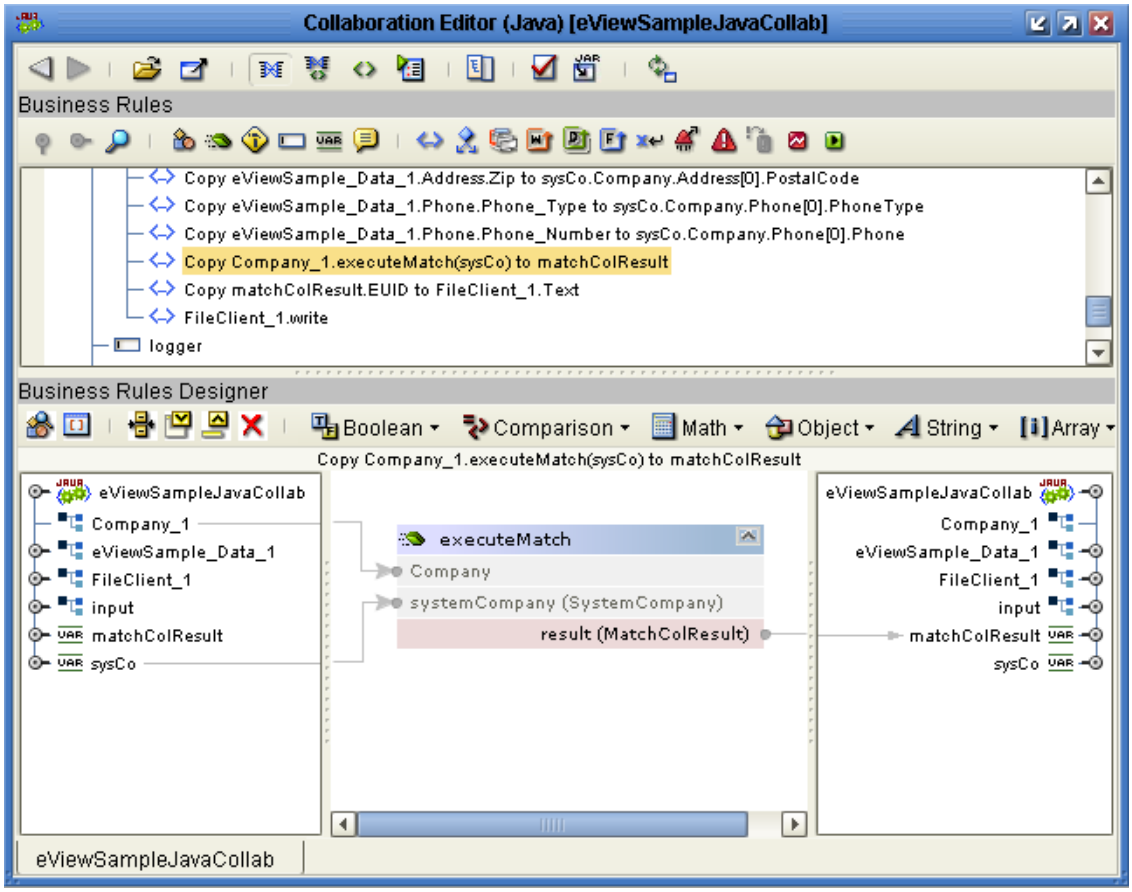
1. Create the Java Collaboration for the Collaboration client Project using the Collaboration Wizard (select the Project, right-click, and then select **New > Collaboration Definition (Java)**).
2. Enter information into the wizard as it applies to the external systems in the Project (for more information,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

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

4. To define custom processing using the eView Studio API, do the following using the Collaboration Editor (Java).
   - A. In the left pane of the Business Rules Designer, right-click the eView Studio 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 32 illustrates a sample of the **executeMatch** method in the Collaboration Editor.

Figure 32 eView Studio Method OTD in Collaboration Editor



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

Creating the Collaboration Client Project Connectivity Map

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

Note: *Before beginning this procedure make sure an external application component is defined for an application that sends information to the master index (source system) and optionally for an application that receives information from the master index (destination system). For testing purposes, you can use File eWays instead of external application eWays for the source and destination systems.*

To create a Collaboration client 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 Change the name of the Connectivity Map.

A Click the **Connectivity Map** icon in the Project Explorer.

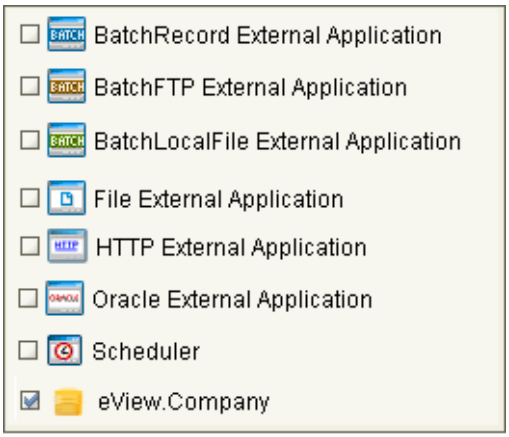
B Change the default name to the name you want to use.

C On the confirmation message, click **OK**.

5 On the Connectivity Map Editor toolbar, click the down arrow next to the **External Application** icon and select the check box next to the name of the External Application that will send messages to eView Studio.

The External Application icon appears in the Connectivity Map Editor toolbar.

Figure 33 External System Menu



6 Drag the External Application icon from the Connectivity Map Editor toolbar to the canvas.

7 If your data flow includes a destination External Application, repeat steps 4 and 5 for the destination application, placing the icon to the far right of the source External Application icon.

8 Drag the Java Collaboration that references the eView Studio method OTD onto the Connectivity Map Editor canvas between the source and destination External Applications.

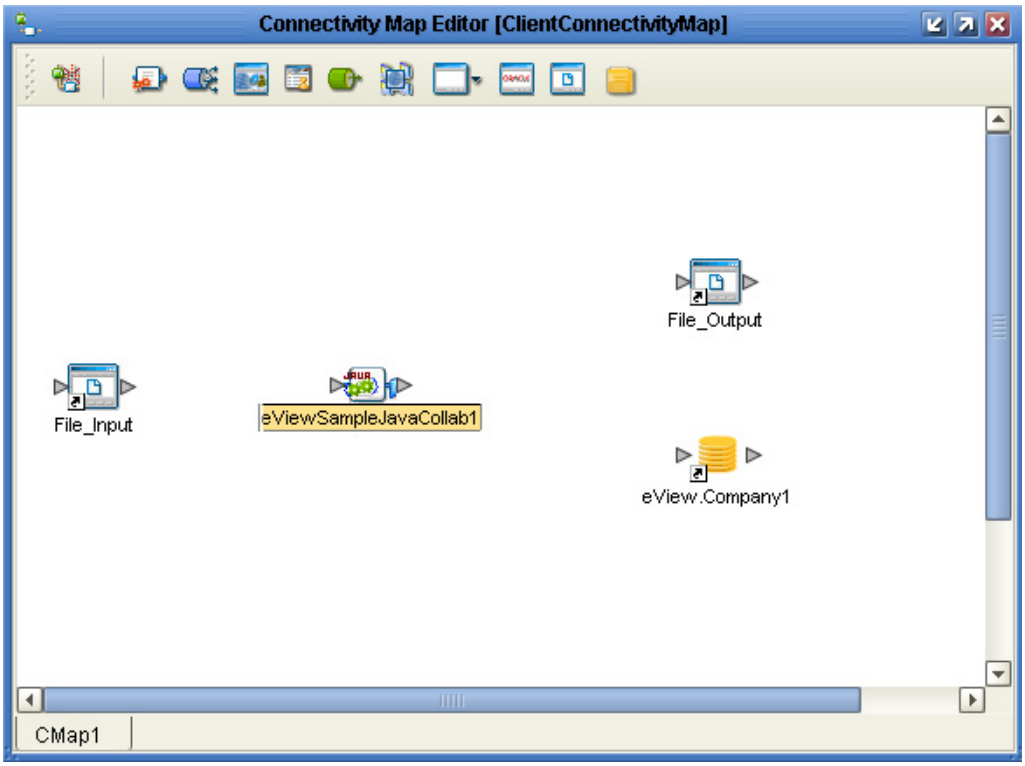
9 On the Connectivity Map Editor toolbar, click the down arrow next to the **External Application** icon and select the check box next to the name of the eView Studio Application you want to integrate.

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

10 Drag the eView Studio application icon from the Connectivity Map Editor toolbar onto the canvas to the lower right of the Collaboration icon.

The Connectivity Map should now look similar to Figure 34.

Figure 34 Collaboration Client Connectivity Map



11 Save the Connectivity Map.

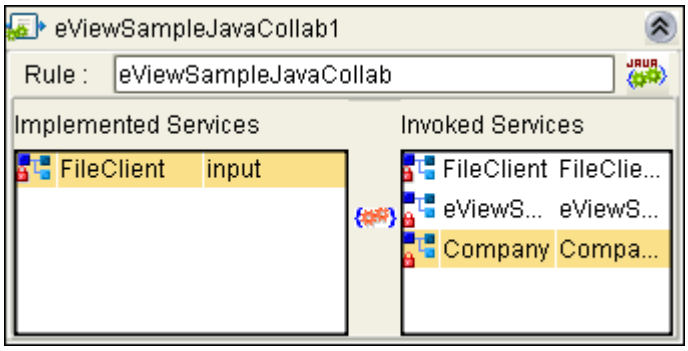
Connecting Connectivity Map Components

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 Collaboration Client Project Connectivity Map" on page 115**.

To connect Connectivity Map components

1 In the Connectivity Map, double-click the **Service** icon to display the Service Binding dialog, as shown in Figure 36.

Figure 35 Collaboration Client - Service Binding Dialog

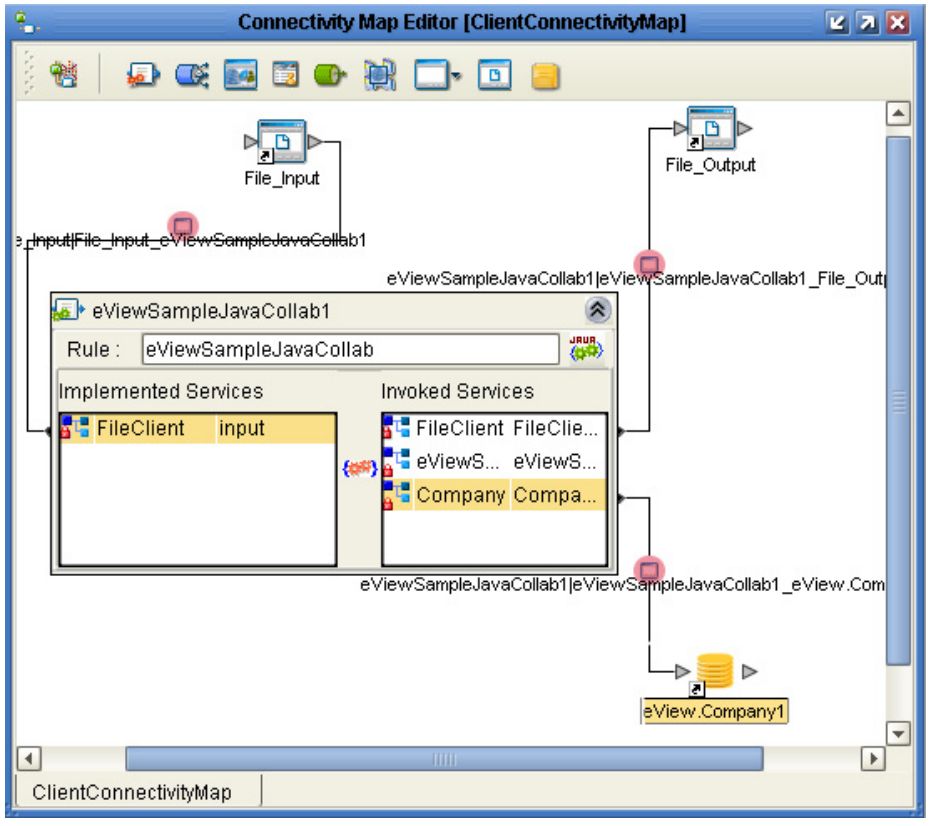


2 Drag the source system from the Implemented Services box in the Service Binding dialog to the external source system icon on the canvas.

3 Drag the eView Studio application from the Invoked Services box in the Service Binding dialog to the eView Studio application icon on the canvas.

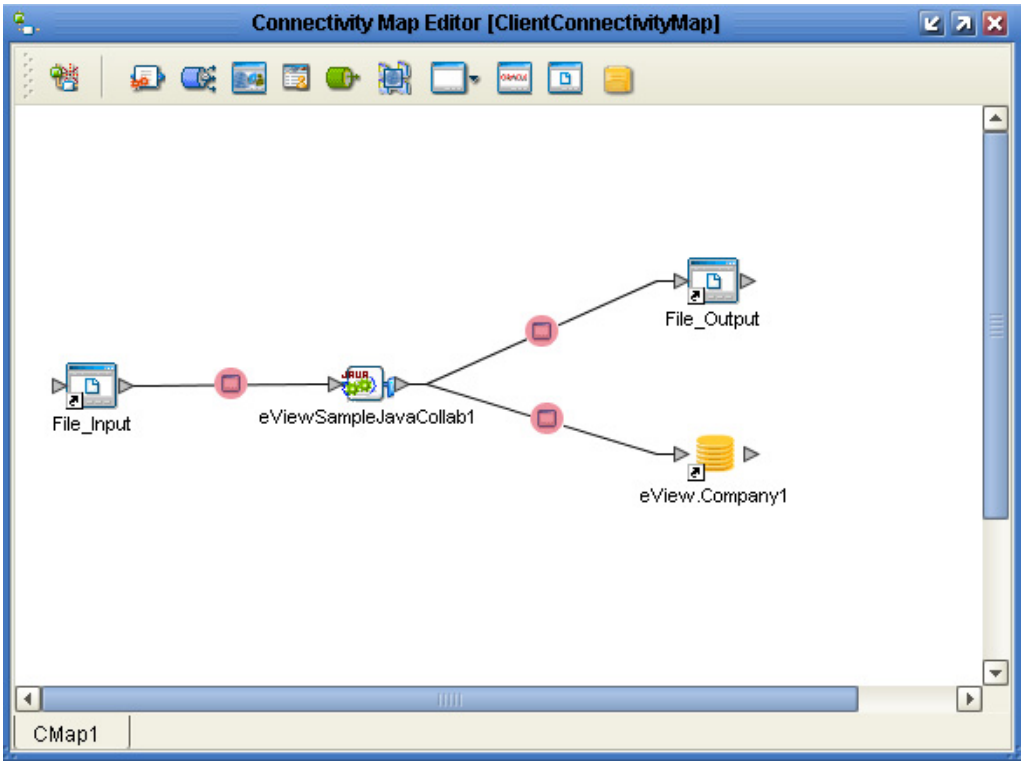
4 If you defined a destination system, drag the appropriate service from the Invoked Services box in the Service Binding dialog to the external destination system icon on the canvas.

Figure 36 Collaboration Client - Service Binding Dialog Connections



5 Close the Service Binding dialog. The Connectivity Map connections should look similar to Figure 37.

Figure 37 Collaboration Client Connectivity Map With Connections



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

7 Double-click the icon between the Service and the eView Studio application to remove the red warning circle. (No configuration is required.)

8 Save and close the Connectivity Map.

Incorporating the JMS Topic into the Connectivity Map

If you defined a JMS Topic in the eView Studio server Connectivity Map, you must add the topic to the Collaboration client Connectivity Map in order to publish the messages to external systems. This involves adding the JMS Topic and associated components to the Connectivity Map and configuring the Collaboration for the connected Service.

Before beginning this procedure, make sure the server Project contains a JMS Topic (this is described in **"Creating the eView Studio Server Connectivity Map" on page 110**

To add the JMS Topic to the Connectivity Map

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

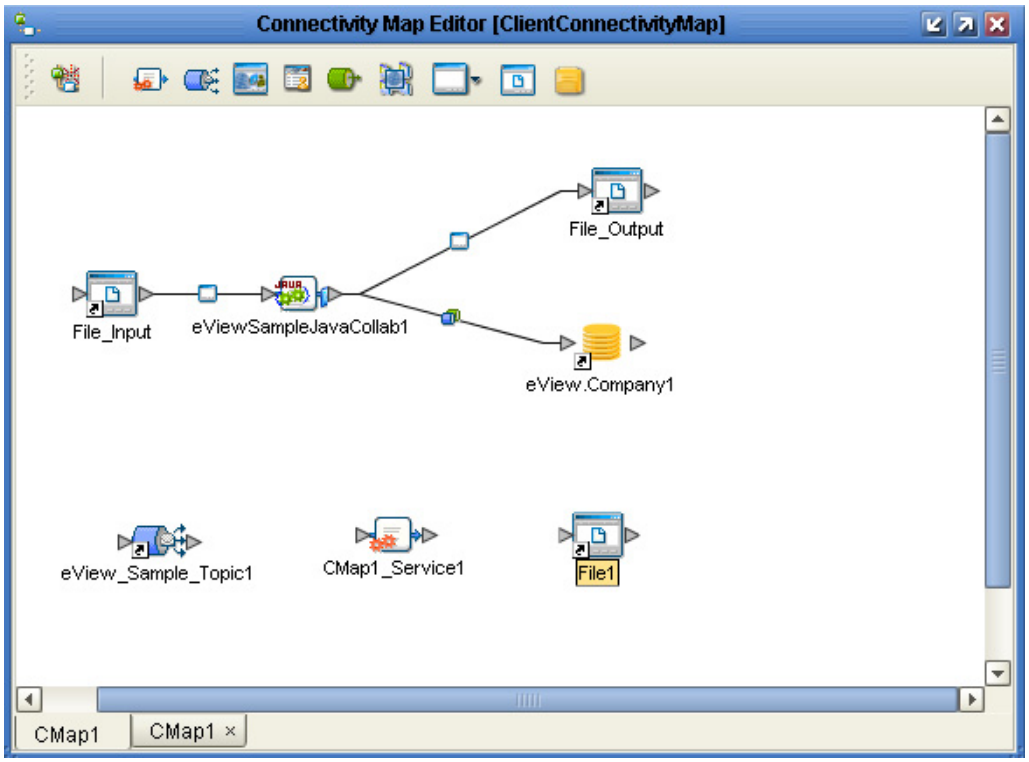
2 Drag the JMS Topic from the eView Studio server Project (in the Project Explorer pane) to the Connectivity Map Editor and place it below the existing Connectivity Map components.

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

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

The Connectivity Map should now look like Figure 38.

Figure 38 Collaboration Client Connectivity Map with JMS Topic

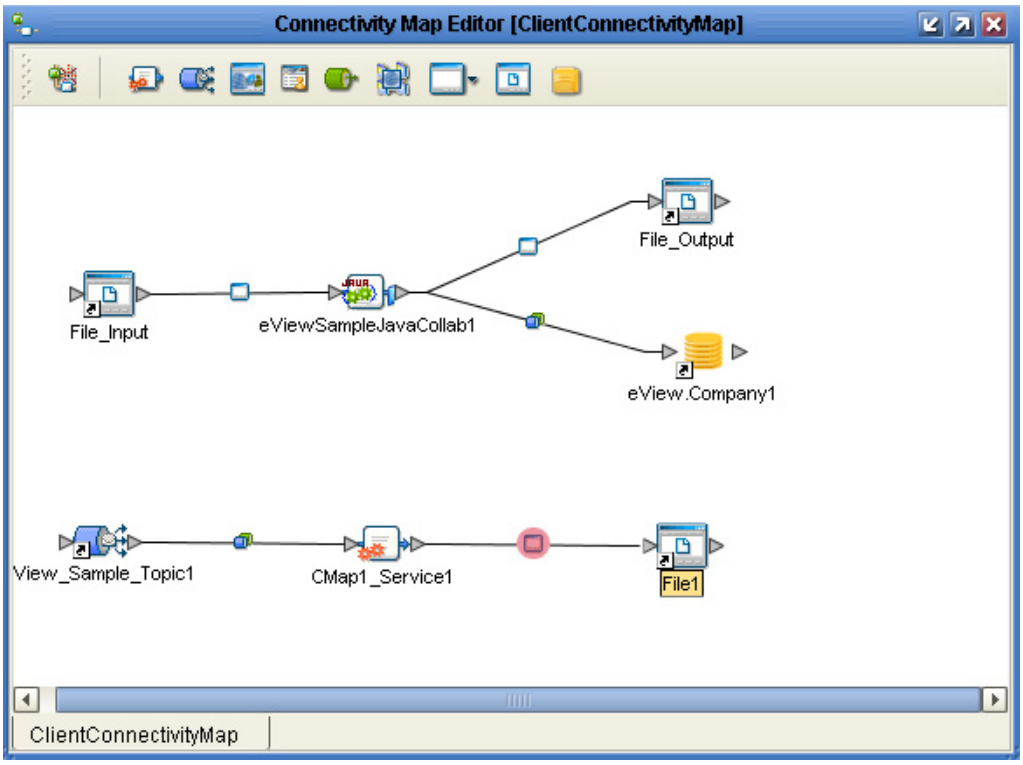


5 In the Connectivity Map Editor, place the cursor over the arrow to the right of the **Topic** icon until the cursor turns into a hand, and then drag it into the Service to connect the two objects.

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

The Connectivity Map should now look similar to Figure 39.

Figure 39 Collaboration Client Connectivity Map with JMS Topic Connections



7 Configure the JMS Client Connection (for more information, see the *Sun SeeBeyond eGate Integrator JMS Reference Guide*).

8 Double-click the External Application eWay to configure the location and parameter settings (see the appropriate eWay user's guide for more information).

9 Save the Connectivity Map, and continue to **"Configuring the Outbound Collaboration"**.

Configuring the Outbound Collaboration

Once you create the JMS Topic in the Connectivity Map, you must configure the Java Collaboration that processes messages from the eView Studio 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 119**.

To configure the outbound Collaboration

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

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

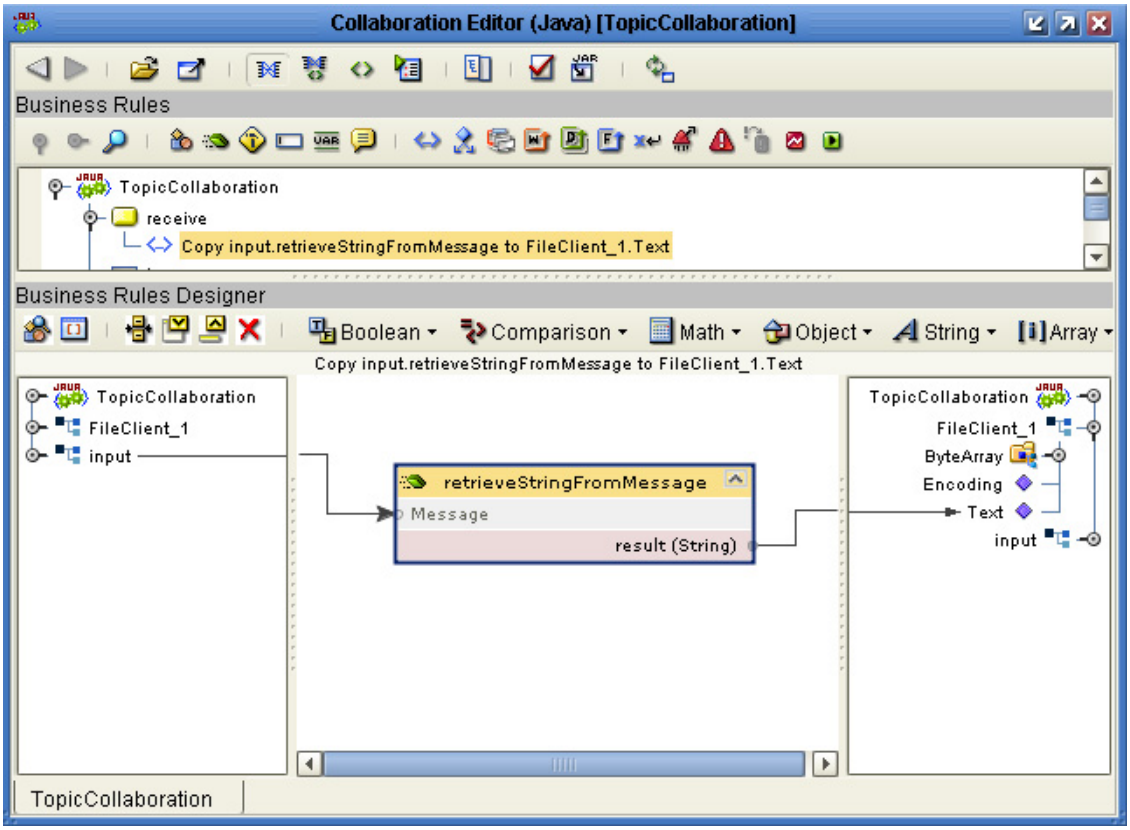
3 Enter information into the Collaboration Definition Wizard, with the following guidelines:

- For the **Web Service Type**, select the existing JMS receive type (navigate to **Sun SeeBeyond\eGate\JMS** and select **receive**).

- Select the appropriate outbound OTD for the external systems in the Project (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 40).

Figure 40 Outbound Java Collaboration for JMS Topic



5 Save the Collaboration to the Repository.

6 Open the Collaboration client Connectivity Map, and then 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 Client Connectivity Components

In eInsight client Projects, the Connectivity Map contains business logic and information about how data is transferred between the master index and external systems through an eInsight Business Process. The Business Process defines how data is transformed before being sent to the master index. This section describes how to incorporate eView Studio methods into a Business Process, create the Connectivity Map, and then add and connect the connectivity components. Perform the following tasks to define connectivity components for Business Processes.

- **Including eView Studio Methods in a Business Process** on page 123

- **Connecting the Business Process Components** on page 124
- **Creating the eInsight Client Connectivity Map** on page 125
- **Connecting Connectivity Map Components** on page 127

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

Including eView Studio Methods in a Business Process

You can process data through a Collaboration, an eInsight Business Process, or both. If you are processing data by both methods, be sure the processing logic is the same for both. Before including the eView Studio application in the Connectivity Map for the eInsight client Project, you must add eView Studio methods to a Business Process. This section provides instructions for adding methods. For more information about the available methods, see the *Sun SeeBeyond eView Studio Reference Guide*.

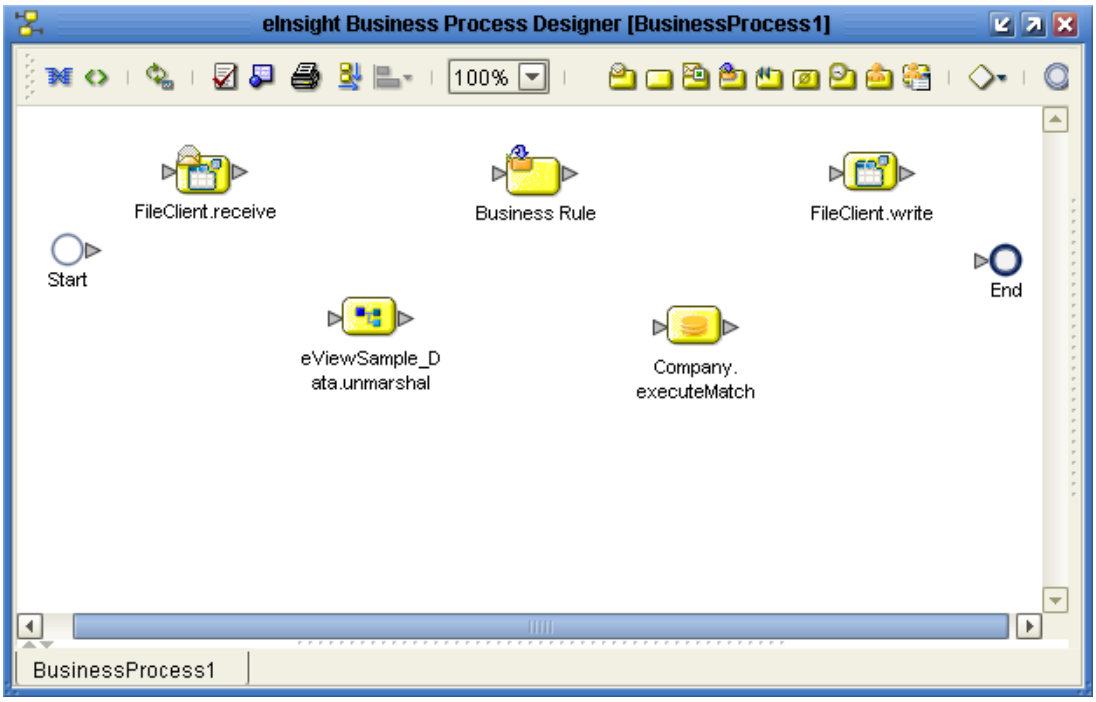
To include eView Studio methods in a Business Process

1. Plan and design a new Business Process.
2. In the Project Explorer pane of Enterprise Designer, right-click the name of the eInsight client Project, point to **New**, and then select **Business Process**.
3. Select the name of the Business Process and rename it.
4. In the Business Process Designer, create the components of the Business Process (for more information, see the *Sun SeeBeyond eInsight Business Process Manager User's Guide*).
5. For each eView Studio method to include, do the following:
   A. In the eView Studio server Project, expand the method OTD folder to display the method list.
   B. Drag the method you want to use into the Business Process Designer.

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

Figure 41 on page 124 illustrates a Business Process with the **executeMatch** eView Studio method.

Figure 41 eInsight Business Process with eView Studio Activity



6 Save the Business Process and continue to the next step, **"Connecting the Business Process Components"**.

Connecting the Business Process Components

This section describes how to connect the components of a Business Process that incorporates eView Studio methods. Make sure you have completed all of the steps in **"Including eView Studio Methods in a Business Process" on page 123** and have the Business Process open in the eInsight Business Process Designer.

To connect the Business Process Components

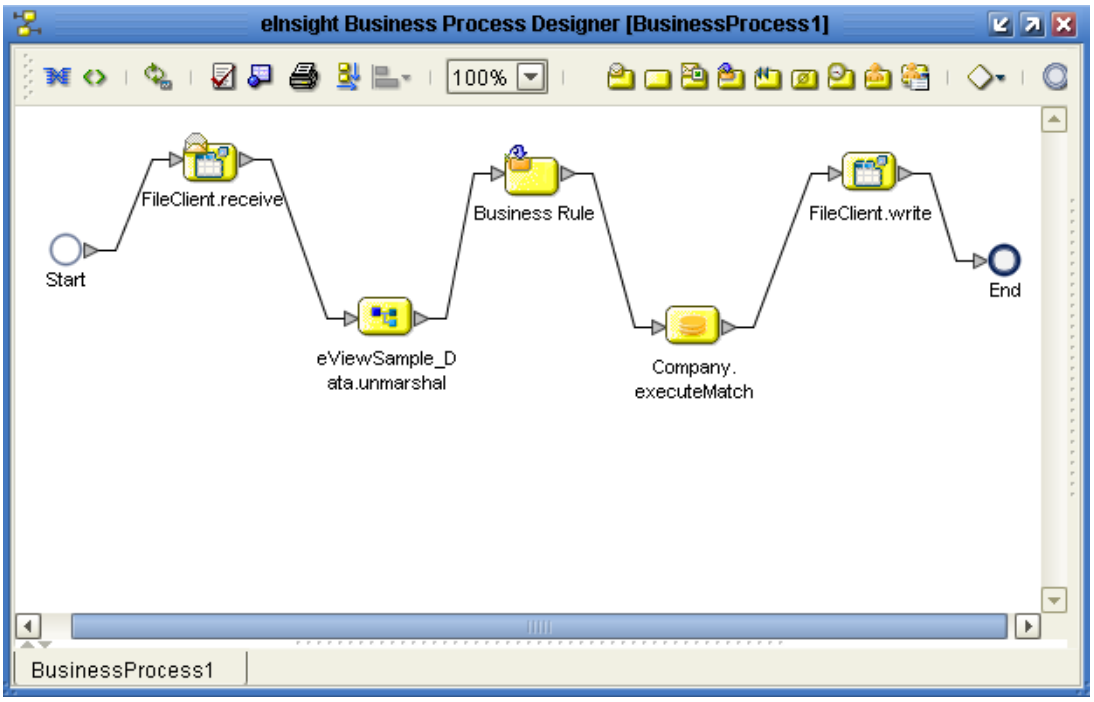
1 In the Business Process Designer, place the cursor over the arrow to the right of the **Start** icon until the cursor turns into a hand.

2 Click the arrow and drag it to the first Business Process component.

3 Follow the same procedure to link each activity in the order in which they should be processed.

The business process should look similar to Figure 42.

Figure 42 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 *Sun SeeBeyond eInsight Business Process Manager User's Guide*.)

5 Create any additional processing logic as needed.

6 Save the Business Process to the Repository.

Creating the eInsight Client Connectivity Map

Connectivity between the eView Studio application and eInsight BPM is defined in the Connectivity Map of the eInsight client Project. This section describes how to create and configure the eInsight client Connectivity Map. You can optionally include a JMS Topic (for information and instructions, see **"To add the JMS Topic to the Connectivity Map" on page 119**). You only need to incorporate the topic if you added a JMS Topic to the server Project and if you want to publish eView Studio messages to external systems.

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

To create the eInsight client 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 Change the name of the Connectivity Map.

   A Click the **Connectivity Map** icon in the Project Explorer.

   B Change the default name to the name you want to use.

   C On the confirmation message, click **OK**.

5 Add External Applications:

   A On the Connectivity Map Editor toolbar, click the down arrow next to the **External Application** icon.

   B Select the check box next to the name of the External Application that will send messages to the master index (see **Figure 33 on page 116**).

   The External Application icon appears in the Connectivity Map Editor toolbar.

   C Drag the External Application icon from the Connectivity Map Editor toolbar to the canvas.

   D If your data flow includes a destination External Application, repeat steps 5 and 6 for the destination application, placing the icon to the far right of the source External Application icon.

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

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

8 Add the eView Studio application:

   A On the Connectivity Map Editor toolbar, click the down arrow next to the **External Application** icon.

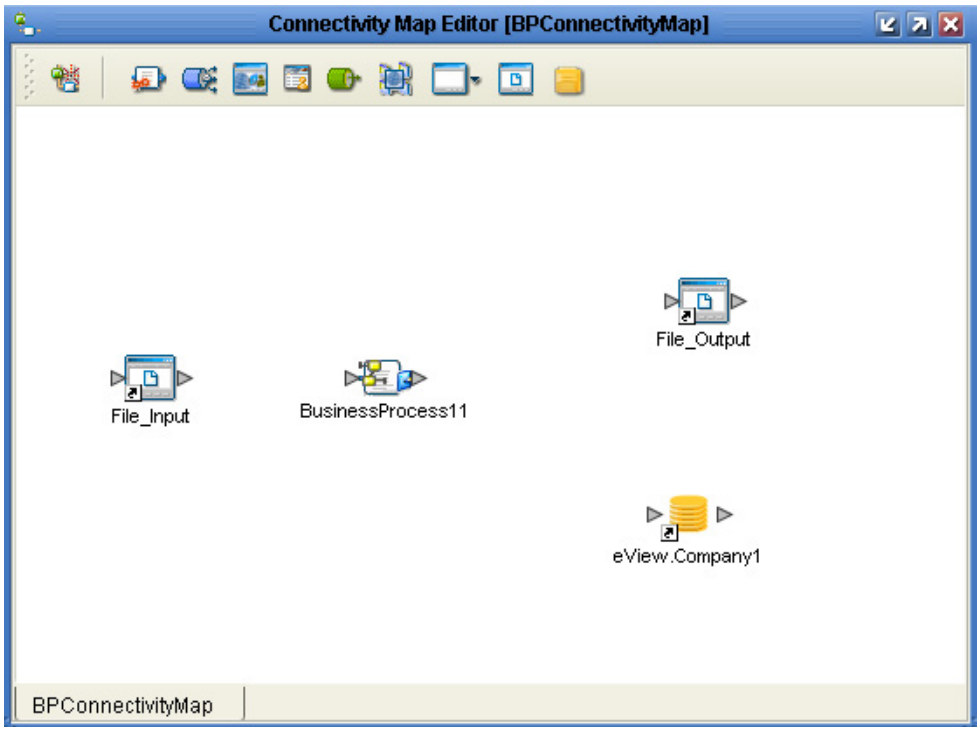
   B Select the check box next to the name of the eView Studio application you want to integrate.

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

   C Drag the eView Studio application icon from the Connectivity Map Editor toolbar onto the canvas to the lower right of the Collaboration icon.

   The Connectivity Map should now look similar to Figure 43.

Figure 43 eInsight Client Connectivity Map



9 Save the Connectivity Map to the Repository.

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

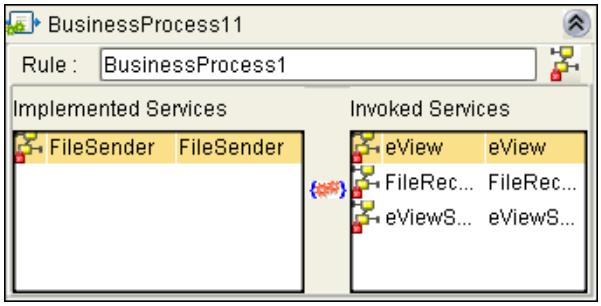
Connecting Connectivity Map Components

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 Client Connectivity Map" on page 125**.

To connect Connectivity Map components

1 In the eInsight Connectivity Map, double-click the **Service** icon to display the Service Binding dialog, as shown in Figure 44.

Figure 44 Service Binding Dialog, eInsight BPM



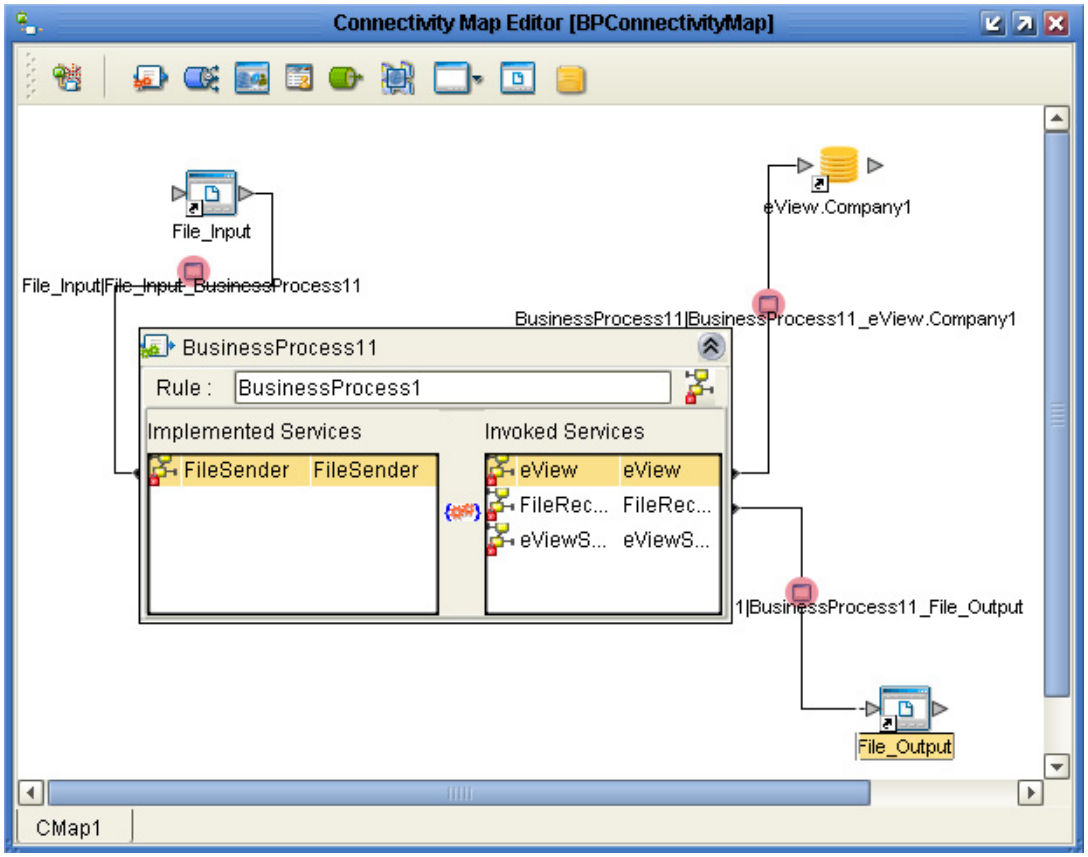
2 Drag the sending application in the Implemented Services box in the Service Binding dialog to the input **External Application** icon on the Connectivity Map Editor.

3 Drag the eView Studio application from the Invoked Services box in the Service Binding dialog to the eView Studio application icon on the Connectivity Map Editor. (This is the service with the same name as the eView Studio application.)

4 Drag the receiving application in the Invoked Services box in the Service Binding dialog to the output **External Application** icon on the Connectivity Map Editor.

Figure 45 illustrates the Connectivity Map with the connections in place.

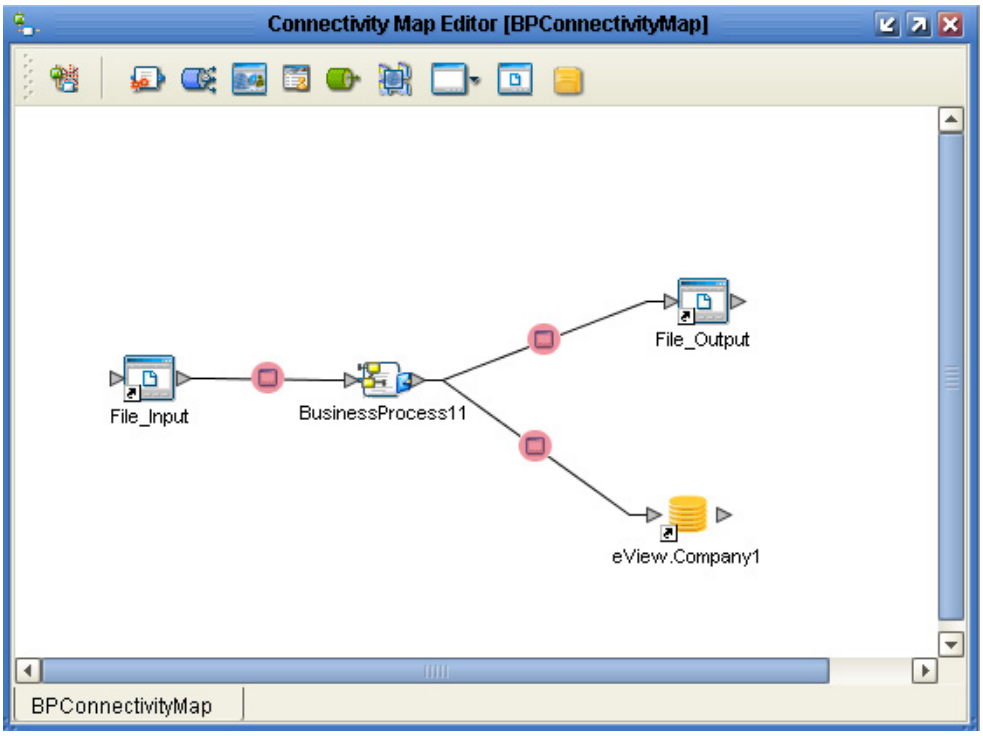
Figure 45 Service Binding Dialog Connections, eInsight BPM



5 Close the Service Binding dialog.

The Connectivity Map window should now look similar to Figure 46.

Figure 46 eInsight Connectivity Map With Connections



6 Click the eWay icon to configure the eWays (for more information, see the user's guide for the type of eWay).

7 Double-click the icon between the Service and the eView Studio application to remove the red warning circle. (No configuration is required.)

8 To add a JMS Topic to the Connectivity Map, follow the instructions under **"Incorporating the JMS Topic into the Connectivity Map" on page 119** and **"Configuring the Outbound Collaboration" on page 121**.

9 Save the Connectivity Map to the Repository.

Chapter 10

Defining the Environment

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

What's in This Chapter

10.1 Environment Components

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

An Environment that supports the eView Studio server Project can include the following components.

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

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

10.2 Building an Environment

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

- **Creating the Environment** on page 131
- **Adding a Logical Host** on page 132
- **Adding Servers** on page 133
- **Adding an External System** on page 135
- **Adding an Oracle External System** on page 136

10.2.1 Creating the Environment

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

To create an Environment

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

4 Select **New Environment** to add an **Environment** icon to the **Environment Explorer** tab.

5 Click twice on the new Environment, enter a unique name for the Environment, and then press **Enter** (see Figure 47).

Figure 47 New Environment

10.2.2 Adding a Logical Host

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

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

To add a Logical Host

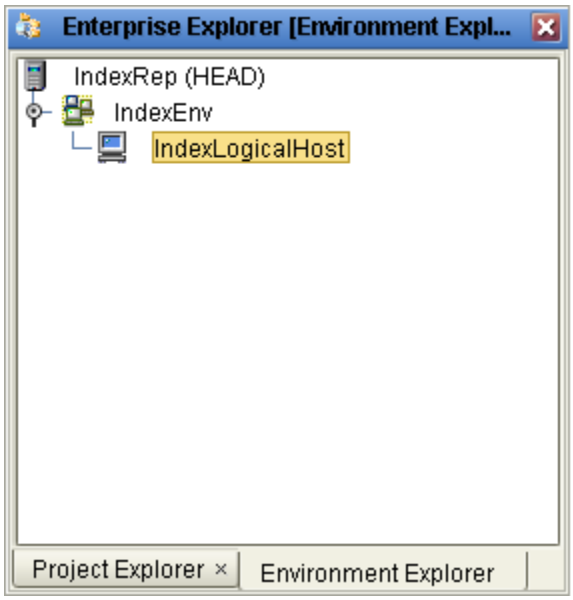
1 Select the **Environment** icon for the new Environment you created.

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

3 Point to **New**, and then select **Logical Host**. A **Logical Host** icon appears on the **Environment Explorer** tab.

4 Click twice on the new Logical Host, enter a unique name for the Logical Host, and then press **Enter** (see Figure 48).

Figure 48 eView Studio Logical Host



10.2.3 Adding Servers

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

Adding a Sun SeeBeyond Integration Server and JMS IQ Manager

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

To add an Integration Server and JMS IQ Manager

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

B Point to **New**, and then select **Sun SeeBeyond JMS IQ Manager**.

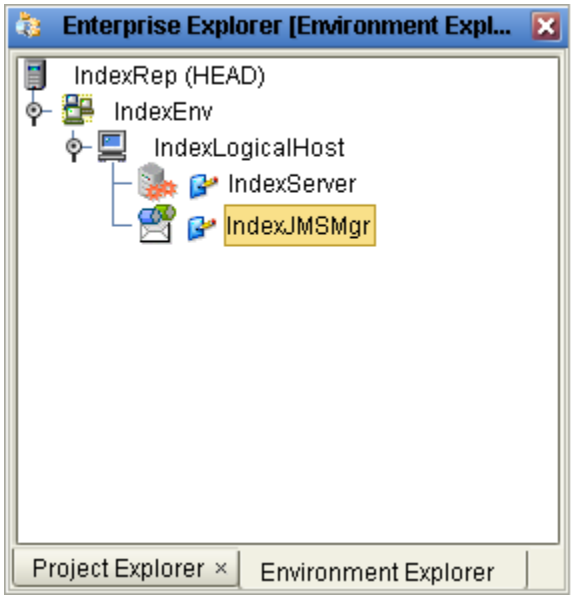
C Modify the name of the IQ manager in the Environment Explorer.

D Right-click the message server, and then select **Properties**.

E Enter the server's URL, administrator login ID, and password.

Figure 49 illustrates an Environment with a Logical Host, Sun SeeBeyond Integration Server, and Sun SeeBeyond JMS IQ Manager.

Figure 49 Integration and JMS Servers



Adding Sun Java System Servers

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

To add Sun Java System servers

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

2 Select the Logical Host icon of the eView Studio Logical Host.

3 To add a Sun Java System Application Server, do the following:

A Right-click the Logical Host icon to display the **Logical Host** context menu.

B Point to **New**, and then select **Sun Java System Application Server**.

C Modify the name of the server in the Environment Explorer.

D Right-click the application server, and then select **Properties**.

E Enter the server's URL, administrator login ID, and password.

4 To add a Sun Java System JMS Server, do the following:

A Right-click the Logical Host icon to redisplay the **Logical Host** context menu.

B Point to **New**, and then select **Sun Java System JMS Server**.

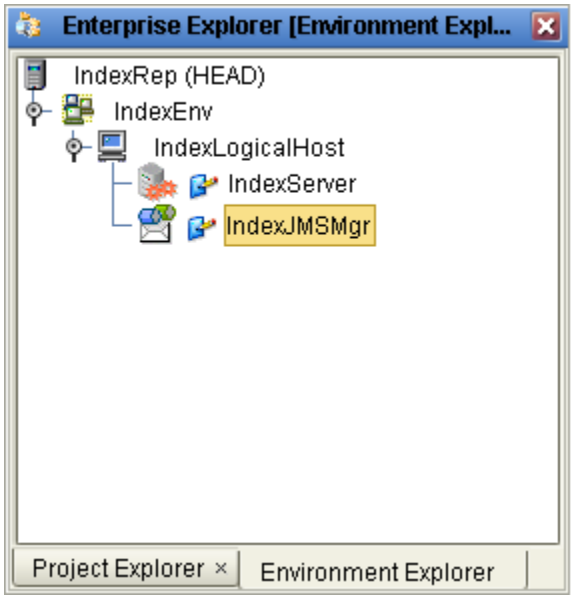
C Modify the name of the IQ manager in the Environment Explorer.

D Right-click the message server, and then select **Properties**.

E Enter the server's URL, administrator login ID, and password.

Figure 50 illustrates an Environment with a Logical Host, Sun Java System Application Server, and Sun Java System JMS Server.

Figure 50 Application and JMS Servers



10.2.4 Adding an External System

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

To add an external system

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

2 Select the eView Studio Environment icon.

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

4 Point to **New**, and then select **<type> External System**, where **<type>** is the type of eWay connecting the external system to eGate (such as Oracle, TCP/IP, and so on).

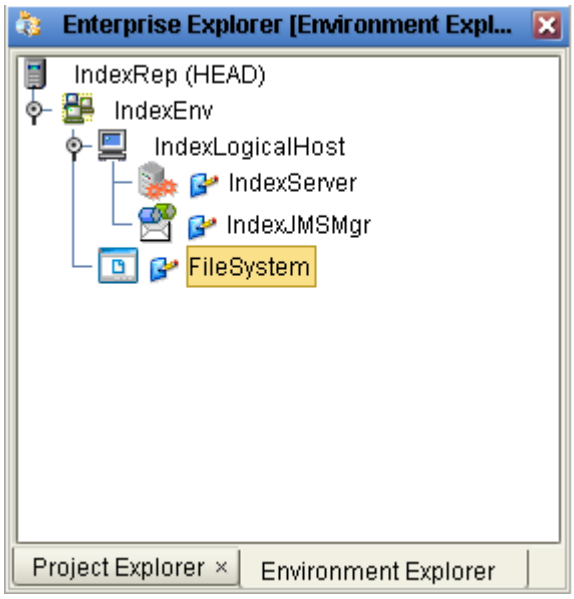
5 In the **External System Name** field, enter the name of the new external system.

6 To configure the external system, right-click the new external system in the Environment, and then select Properties from the context menu.

7 Configure the properties as described in the appropriate user's guide.

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

Figure 51 File External System



10.2.5 Adding an Oracle External System

An Oracle external system is required for eView Studio server Projects that connect to the database using an Oracle eWay. It is required for Projects running on the Sun SeeBeyond Integration Server, but is optional for Projects running on the Sun Java System Integration Server. Perform the following steps to add an Oracle external system to the eView Studio Environment. If you are not using an Oracle eWay, be sure to follow the instructions under **Configuring a Connection Pool Through the Server** on page 137 to create a database connection pool.

To add an Oracle external system

1 In Enterprise Explorer, click the **Environment Explorer** tab and then select the eView Studio Environment icon.

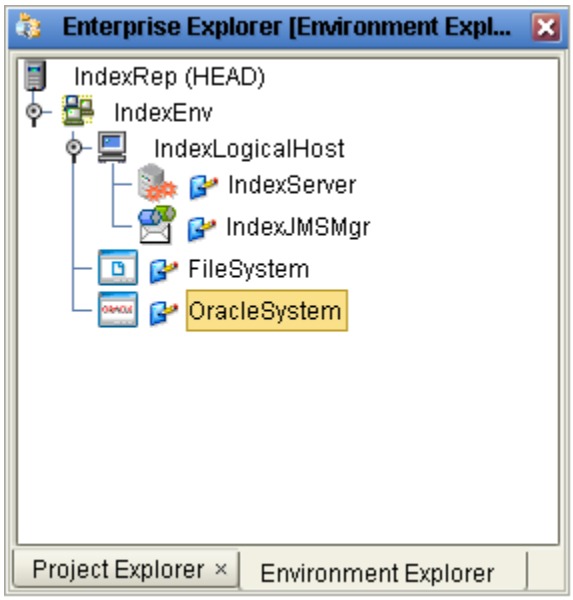
2 Right-click the eView Studio Environment icon to display the context menu.

3 Point to **New**, and then select **Oracle External System**.

4 In the **External System Name** field, enter the name of the new Oracle external system.

5 Click **OK**.

Figure 52 Oracle External System



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

The **Properties** window appears.

7 Expand **Outbound Oracle eWay**, and then select **JDBC Connector Settings**.

8 Define each property in the right portion of the window with information specific to the eView Studio database you created. All fields on the Properties windows are described in the *Sun SeeBeyond eWay Adapter for Oracle User's Guide*. Be sure to only define the properties for the driver type you are using.

Note: *Creating the database is described in* **Chapter 8** *of this guide. Use the information for that database for the properties on this window.*

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

10.3 Configuring a Connection Pool Through the Server

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

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

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

Step 1: Create a JDBC Connection Pool

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

To create a JDBC connection pool

1. In the left portion of the Sun Java System Application Server Admin Console, expand **Resources**, expand **JDBC**, and then select **Connection Pools**.
2. On the Create Connection Pool page, click **New**.
3. Enter values in the fields described in Table 9, and then click **Next**.

Table 9 Connection Pool General Settings

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

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

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

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

Step 2: Create a JDBC Resource

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

To create a JDBC resource:

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

3 Enter the field values described in Table 10.

Table 10 JDBC Resources Fields

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

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

5 Click **OK**.

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 Studio Project and any client Projects that connect the master index to an external system.

What's in This Chapter

11.1 Deployment Overview

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

11.2 Creating a Domain

If eView Studio is running on the IS, make sure you have created and started an instance of the Logical Host containing the IS before defining Deployment Profiles for your Projects. This section describes how to create a domain using the Domain Manager. You can also create the instance using a command-line tool. The command-

line method is described in the *Sun SeeBeyond eGate Integrator System Administration Guide*.

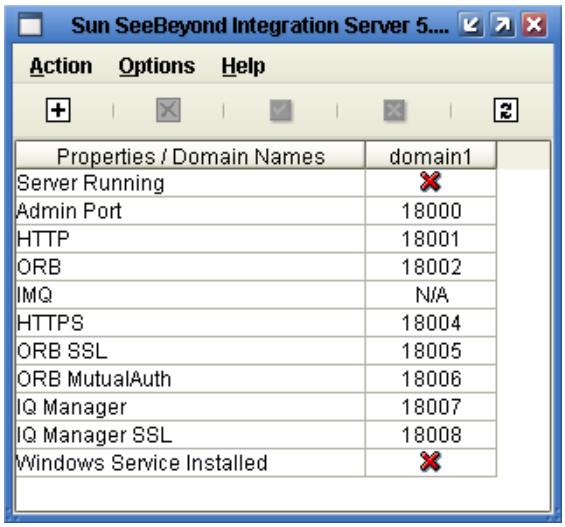
Important: *The instructions below only pertain to applications running on the Sun SeeBeyond Integration Server. For applications running on the Sun Java System Application Server, refer to you Sun documentation for information about creating and starting a domain. Before starting the Logical Host, make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

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

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

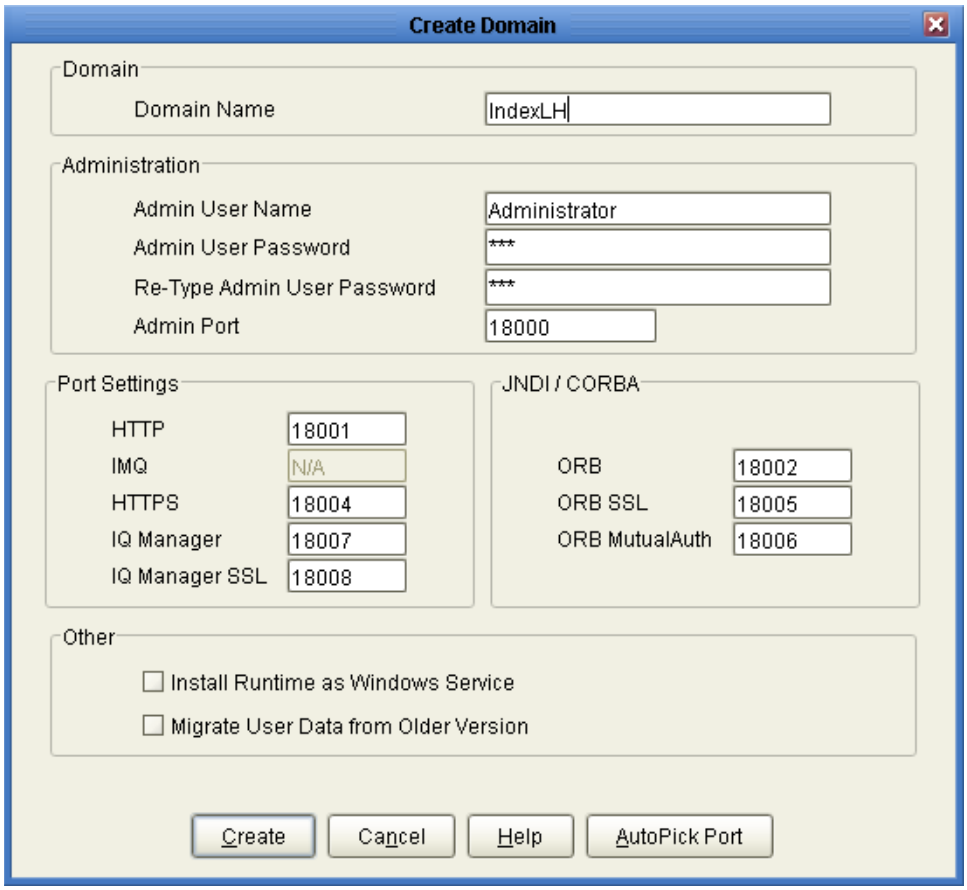
   The Domain Manager appears.

Figure 53 Domain Manager



3. On the Domain Manager toolbar, click the **Create a New Domain** tool. The **Create Domain** dialog box appears (see Figure 54).

Figure 54 Create Domain Dialog Box



4 You can change the default values of any of the fields listed in Table 11.

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

Table 11 Fields in Create Domain Dialog Box

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

Table 11 Fields in Create Domain Dialog Box

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

5 If you want to install the SeeBeyond Integration Server as a Windows service, select the **Install Runtime as Windows Service** check box. The service name will be **IS 5.1 <domain_name>**.

6 Click **Create**.

7 When the **Message** dialog box indicates that the domain has been successfully created, click **OK**.

8 (Make sure the eView Studio database is running before performing this step.) On the Domain Manager window, select the new domain and then click **Start an Existing Domain**.

9 When the **Message** dialog box indicates that the domain has been successfully started, click **OK**.

11.3 Defining Deployment Profiles

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

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

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

name of the eView Studio Project, <deployment_profile> is the name of the Project's Deployment Profile, and <integration_server> is the name of the server on which the application is deployed). The file is named ***<project><deployment_profile>.ear****; for example,* ***eViewPersonIndexDeploy.ear****).*

11.3.1 Deploying the eView Studio Server Project

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

- **Creating the Deployment Profile** on page 144
- **Mapping Project Components to Environment Components** on page 146
- **Building and Deploying the Server Project** on page 148

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

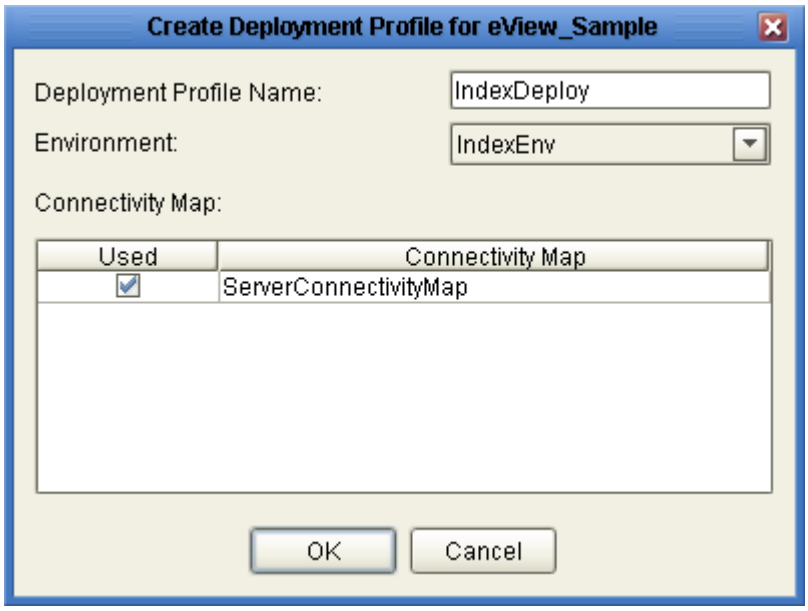
Creating the Deployment Profile

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

To create an eView Studio application Deployment Profile

1. In Enterprise Explorer, click the **Project Explorer** tab.
2. Select the eView Studio server Project folder.
3. Right-click the mouse to launch the **Project** context menu.
4. Select **New > Deployment Profile** to display the **Create Deployment Profile** dialog shown in Figure 55.

Figure 55 Create Deployment Profile Dialog Box



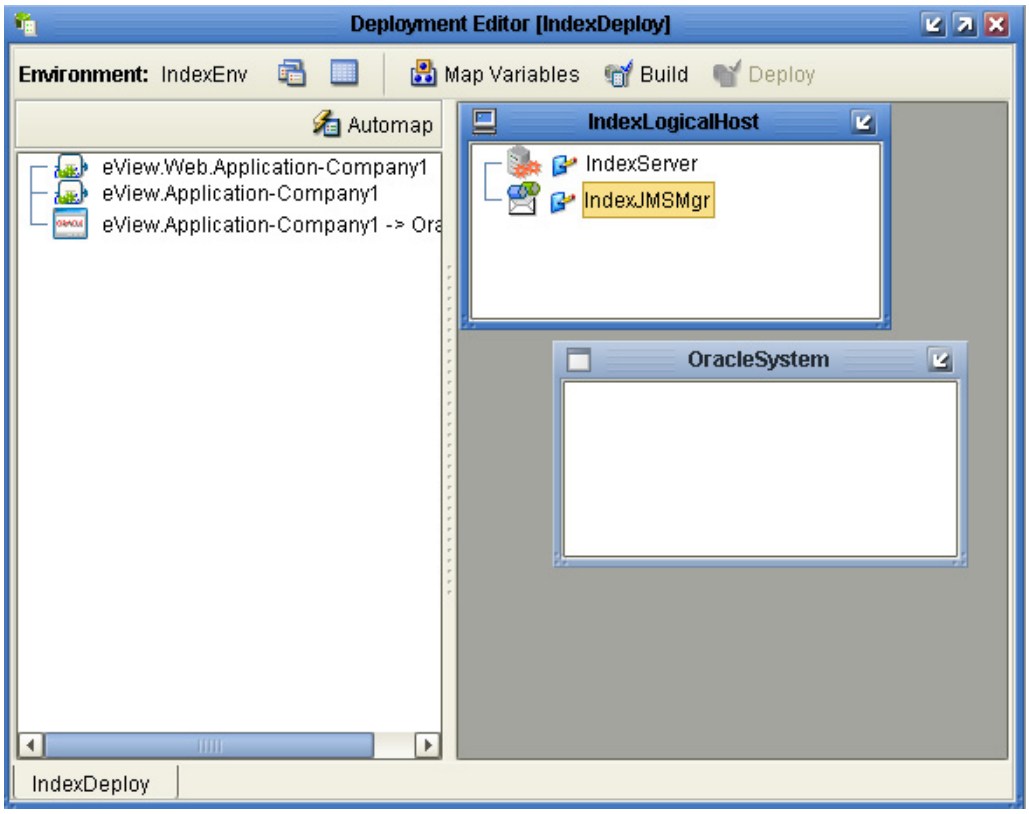
5 In the **Deployment Profile Name** field, type a name for the Deployment Profile.

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

7 In the Connectivity Map table, deselect the Used check box for any Connectivity Maps you do not want to include in the deployment.

8 Click **OK** to add a **Deployment Profile** icon to the eView Studio Project and display the Deployment Editor window, shown in Figure 56.

Figure 56 Deployment Editor - Server Project



9 Continue to **"Mapping Project Components to Environment Components"**.

Mapping Project Components to Environment Components

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

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

To map eView Studio Project components manually

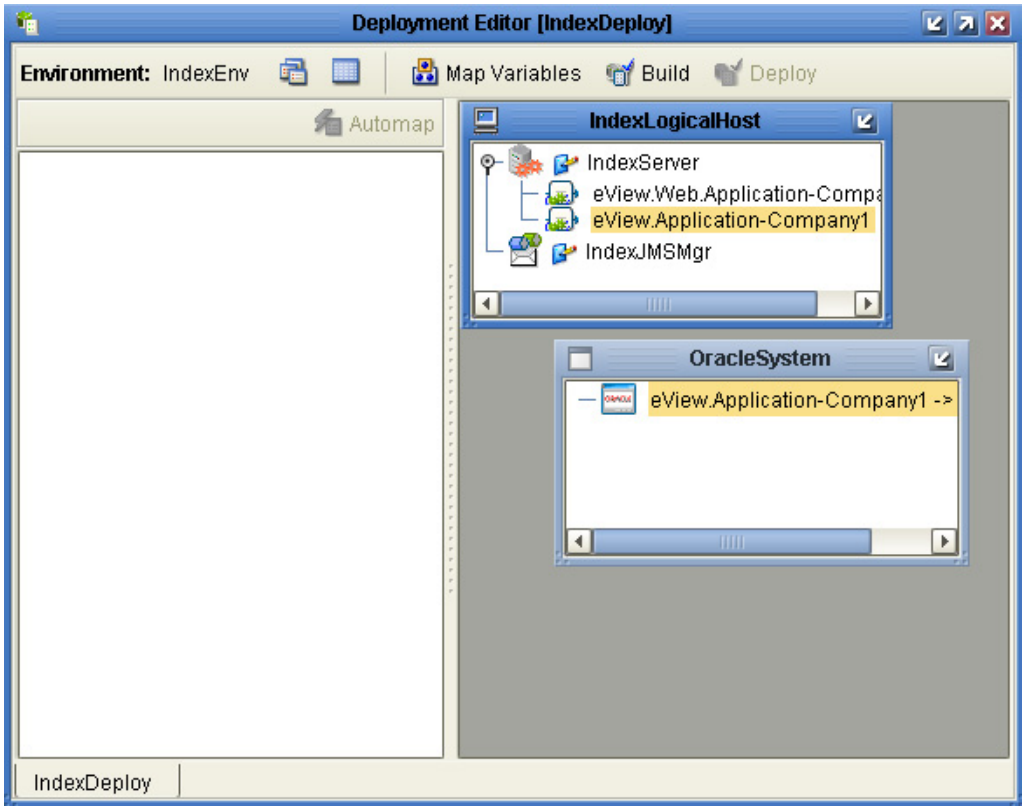
1 With the eView Studio Project Deployment Profile open in the Deployment Editor, drag the **eView.Web.Application-<application_name>** and **eView.Application-<application_name>** icons onto the integration or application server in the Deployment Editor (the server appears in the Logical Host).

2 If you defined a JMS Topic to publish the master index messages, drag the **JMS Topic** icon onto the JMS IQ Manager in the Logical Host.

3 If the server Project contains an Oracle eWay, drag the **Oracle External Application** icon onto the Oracle External System in the right pane.

Figure 57 illustrates the updated Deployment Profile.

Figure 57 Mapped eView Studio Components in the Deployment Editor



4 Continue to **"Building and Deploying the Server Project"**.

To automatically map eView Studio Project Components

1 With the eView Studio Project Deployment Profile open in the Deployment Editor, click **Automap**.

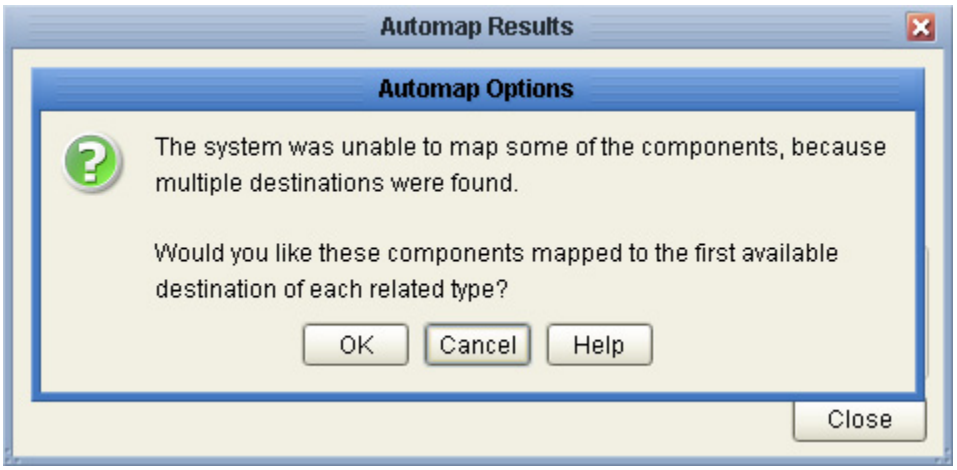
The Automap Results dialog appears.

2 Review the results of the mapping on the Automap Results dialog. If the appropriate container for the component cannot be found, the component remains unmapped.

3 If more than one container was found for a component, click **Automap Option** to review mapping options and do one of the following:

- To accept the Automapping option, click **OK**.
- To map the components manually, click **Cancel**.

Figure 58 Automap Options Dialog



4 To map any components that could not be automatically mapped, follow the instructions under **To map eView Studio Project components manually** on page 146.

5 Continue to **"Building and Deploying the Server Project"**.

Building and Deploying the Server Project

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

To build and deploy the server Project

1 Click **Build** to generate the Project files.

2 After the build process finishes successfully, click **Deploy** to deploy the Project to a server.

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

3 Define EDM users, as described under **"Defining Security" on page 155**.

11.3.2 Deploying the Collaboration Client Project

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

- **Creating a Collaboration Client Deployment Profile** on page 149
- **Mapping Collaboration Client Project Components** on page 150
- **Building and Deploying the Collaboration Client Project** on page 151

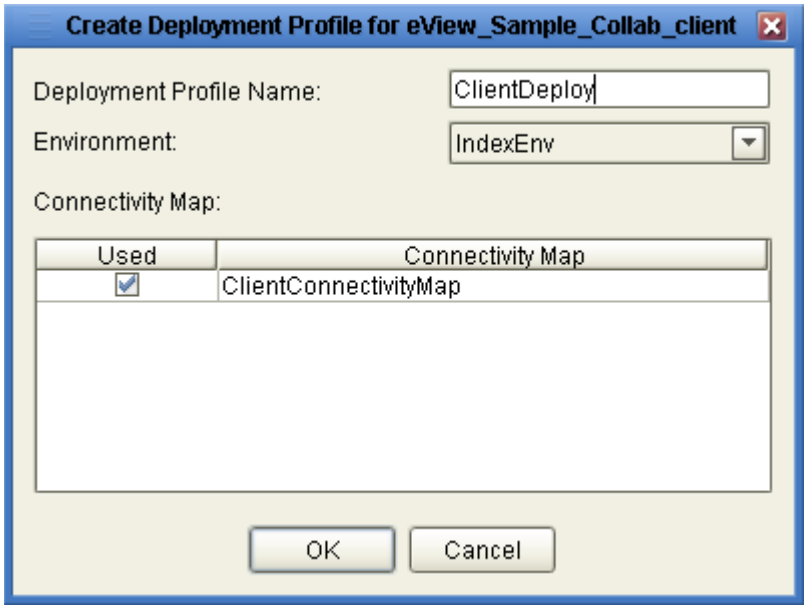
Creating a Collaboration Client Deployment Profile

Before beginning this procedure, you must have created, built, and deployed the eView Studio application Deployment Profile, as described in **"Creating the Deployment Profile" on page 144**. Doing this creates an eView Studio application component in the Environment, which is required for the client deployment.

To create a Collaboration client Deployment Profile

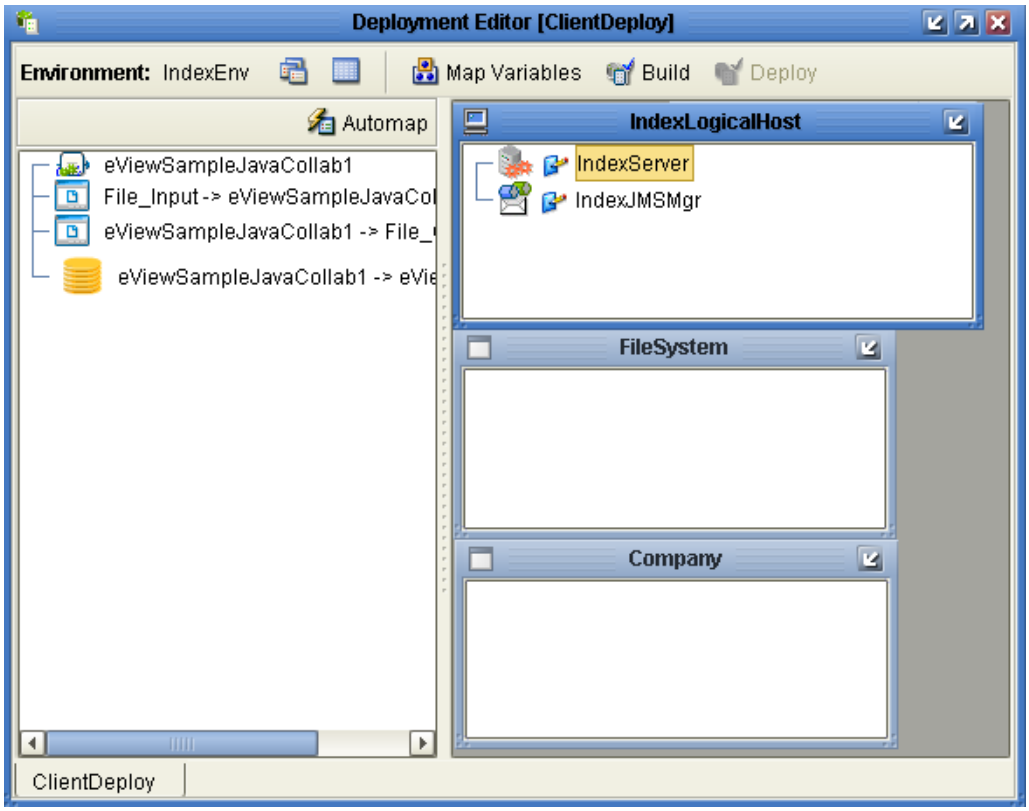
1. In Enterprise Explorer, click the **Project Explorer** tab.
2. Select the Collaboration client Project folder.
3. Right-click the mouse to launch the **Project** context menu.
4. Select **New > Deployment Profile** to display the **Create Deployment Profile** dialog shown in Figure 59.

Figure 59 Create Deployment Profile Dialog Box



5. In the **Deployment Profile Name** field, type a name for the Deployment Profile.
6. In the **Environment** drop-down list, select the Environment you created for the eView Studio Project.
7. Click **OK** to add a **Deployment Profile** icon to the eView Studio Project and display the Deployment Editor window, shown in Figure 60.

Figure 60 Deployment Editor - Client Project



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

8 Continue to **"Mapping Collaboration Client Project Components"**.

Mapping Collaboration Client Project Components

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

Note: *Project components can also be automatically mapped. For more information, see* **Mapping Project Components to Environment Components** *on page 146 and* **To automatically map eView Studio Project Components** *on page 147.*

To map Collaboration client Project components

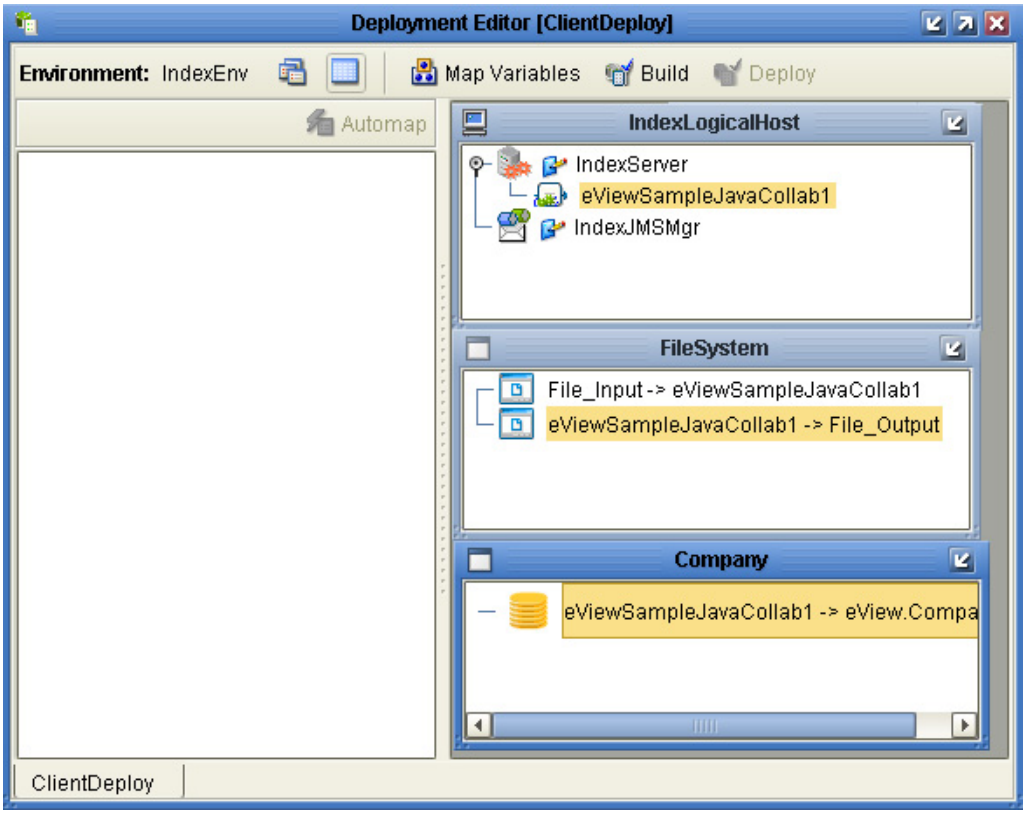
1 With the Collaboration client Project Deployment Profile open in the Deployment Editor, drag the Service icon(s) onto the integration or application server in the Deployment Editor (the server appears in the Logical Host).

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

3 Drag the external system eWays to the appropriate external systems in the Environment.

4 If you implemented a JMS Topic, drag the topic to the JMS IQ Manager in the Logical Host.

Figure 61 Mapped Client Components in the Deployment Editor



5 Continue to **"Building and Deploying the Collaboration Client Project"**.

Building and Deploying the Collaboration Client Project

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

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

To build and deploy the Collaboration client Project

1 Click **Build** to generate the Project files.

2 Click **Deploy** to deploy the Project to a server.

11.3.3 Deploying the eInsight Client Project

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

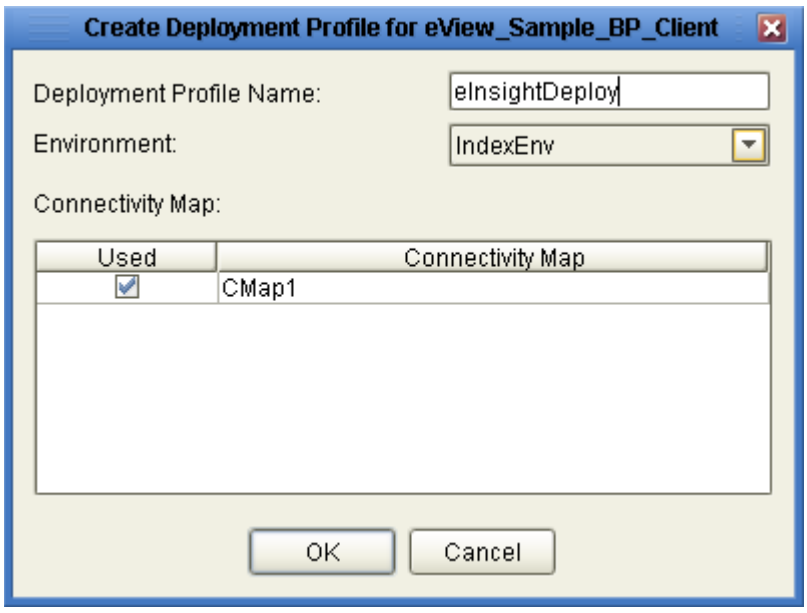
Creating an eInsight Client Deployment Profile

Before beginning this procedure, you must have created, built, and deployed the eView Studio application deployment profile, as described in **"Creating the Deployment Profile" on page 144**. Doing this creates the eView Studio application component in the Environment, which is required for the client deployment.

To create an eInsight client Project Deployment Profile

1. In Enterprise Explorer, click the **Project Explorer** tab.
2. Select the eInsight client Project folder.
3. Right-click the mouse to launch the **Project** context menu.
4. Select **New > Deployment Profile** to display the **Create Deployment Profile** dialog shown in Figure 62.

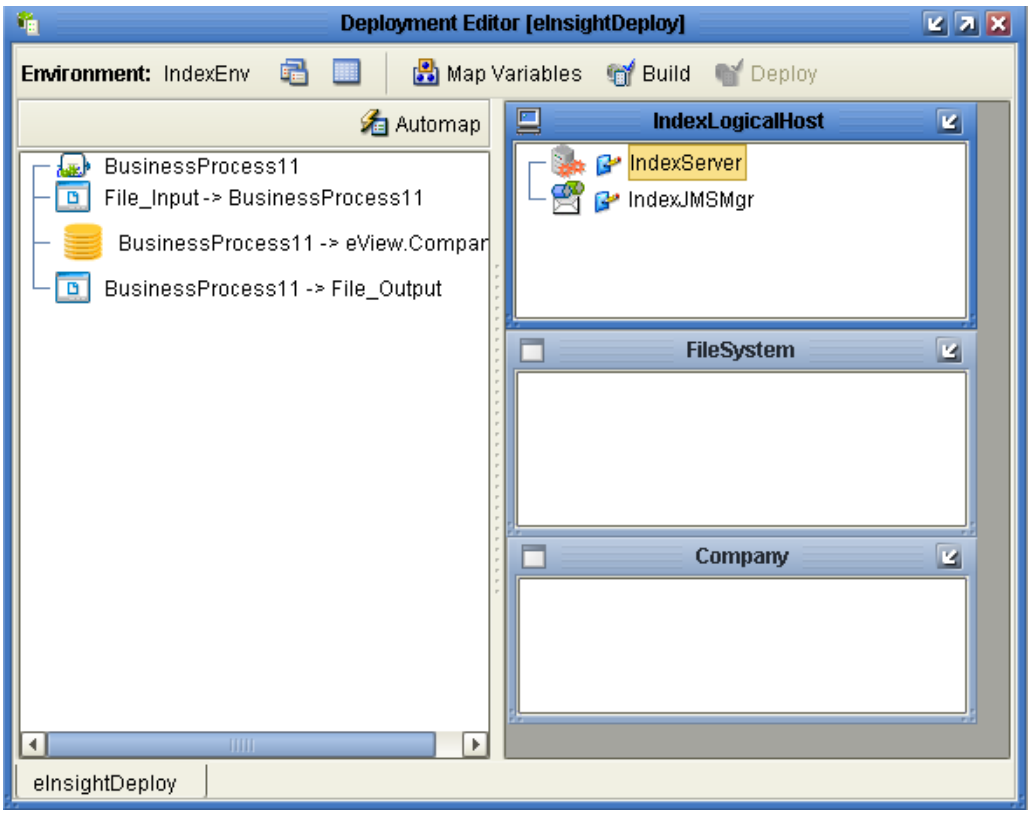
Figure 62 Create Deployment Profile Dialog Box



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

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

Figure 63 Deployment Editor Window



8 Continue to **"Mapping eInsight Client Project Components"**.

Mapping eInsight Client Project Components

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

Note: *Project components can also be automatically mapped. For more information, see* **Mapping Project Components to Environment Components** *on page 146 and* **To automatically map eView Studio Project Components** *on page 147.*

To map eInsight client Project components

1 With the eInsight client Project Deployment Profile open in the Deployment Editor, drag the Service/Business Process icon onto the integration or application server in the Deployment Editor (the server appears in the Logical Host).

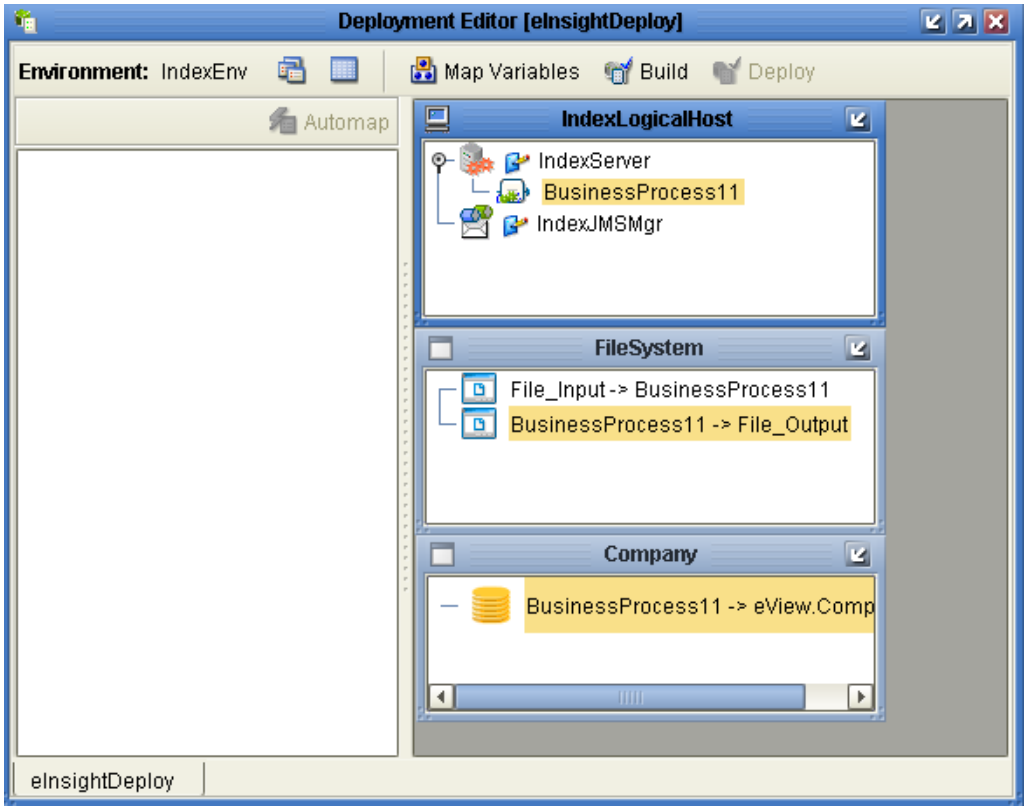
2 Drag the eView Studio application icon onto the eView Studio application deployment component (this is named after the eView Studio application).

3 Drag the external system eWays to the appropriate external systems in the Environment.

4 If you implemented a JMS Topic, drag the topic to the JMS IQ Manager in the Logical Host (not shown).

Figure 64 illustrates the updated Deployment Profile.

Figure 64 Mapped Client Components in the Deployment Editor



5 Continue to **"Building and Deploying the eInsight Client Project"**.

Building and Deploying the eInsight Client Project

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

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

To build and deploy the eInsight client Project

1 Click **Build** to generate the Project files.

2 Click **Deploy** to deploy the Project to a server.

11.4 Defining Security

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

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

Make sure the domain for the eView Studio application is running before performing this procedure.

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

11.4.1 Defining Security (for the Sun SeeBeyond Integration Server)

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

To connect to the server

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

   The Application Server Deployer page appears with the Manage Servers tab displayed.
3. In the **Add Application Server** section, fill in the fields listed in Table 12.

Table 12 Manager Server Fields

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

Table 12 Manager Server Fields

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

4 Click **Connect to Server**.

5 Once the connection is made, click **Save current user preferences**.

To create an eView Studio user account in the Enterprise Manager

1 Log on to Enterprise Manager.

2 On the Enterprise Manager, expand the J2EE list until you see the server to which you want to add a user account.

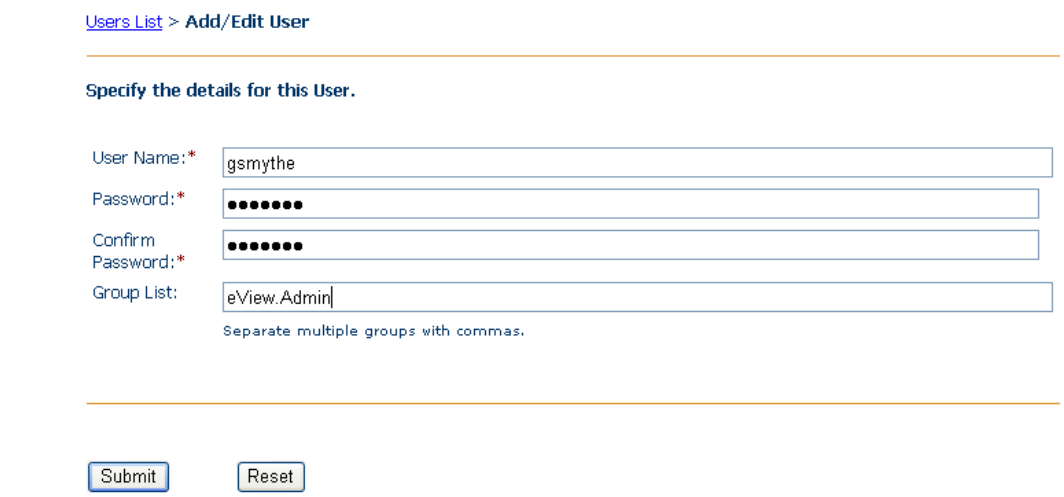
3 Right-click the server, and then click **Manage Integration Server Users**.

The **Users List** window appears.

4 In the **Users List** window, click **Add New User**.

The **Add/Edit User** window appears (see Figure 65).

Figure 65 Add/Edit User Window



5 In the **User Name** field, enter a name for the user.

6 In the **Password** field, enter a password for the user.

7 In the **Confirm Password** field, enter the password again.

8 In the **Group List** field, enter one or more eView Studio user groups, separating multiple groups with a comma. Table 13 lists and describes each eView Studio user group.

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

Table 13 User Groups and Descriptions

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

Table 13 User Groups and Descriptions

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

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

Security for eView Studio running on the Sun Java System Application Server is configured on the Admin Console.

To create an eView Studio user account in the Admin Console

1. Log on to the Sun Java System Application Server Admin Console.
2. In the left portion of the page, expand **Configurations** and then expand the server on which eView Studio is running.
3. Under the server, expand **Security**, expand **Realms**, and then select **file**.
4. On the Edit Realm page, select **Manager Users**.
5. On the File Users page, select **New**.

6 In the **User ID** field, enter a name for the user.

7 In the **Password** field, enter a password for the user.

8 In the **Confirm Password** field, enter the password again.

9 In the **Group List** field, enter one or more eView Studio user groups, separating multiple groups with a comma. Table 13 lists and describes each eView Studio user group.

10 After you have added all required user groups, click **OK**.

Chapter 12

Maintenance Tasks

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

What's in This Chapter

12.1 Archiving Repository Information

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

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

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

12.2 Maintaining the Database

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

12.2.1 Backing up the Database

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

Online Backups

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

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

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

Offline Backups

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

12.2.2 Database Restoration

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

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

12.2.3 Archiving

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

12.3 Monitoring Day to Day Activity

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

- **Enterprise Manager Monitor** on page 162
- **Log Files** on page 163
- **Alerts** on page 164

12.3.1 Enterprise Manager Monitor

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

About the Enterprise Manager Monitor

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

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

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

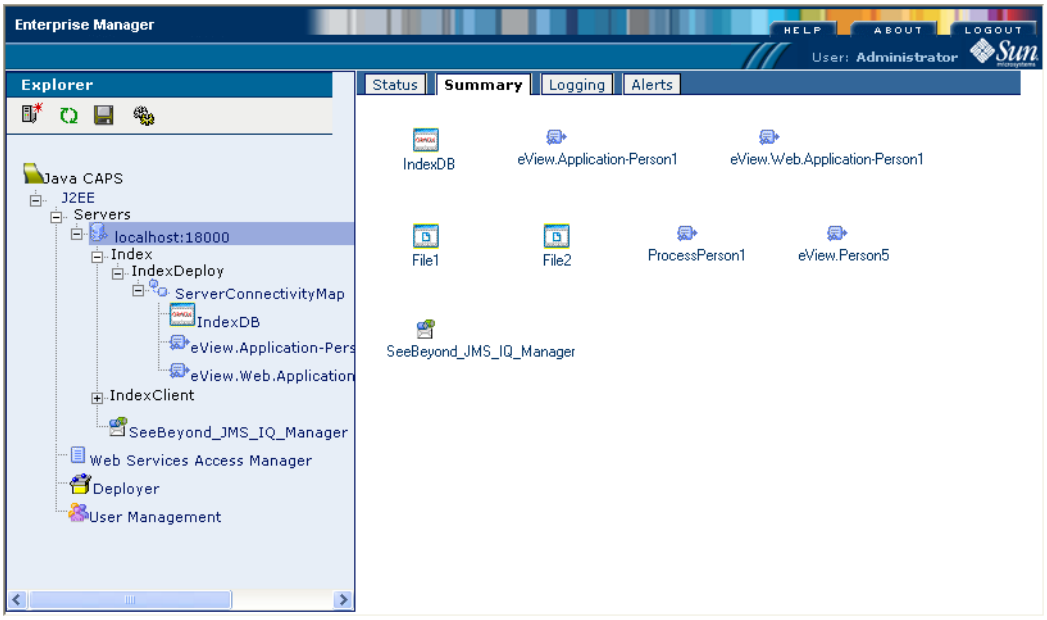
- *Sun SeeBeyond eGate Integrator System Administration Guide*
- *Sun SeeBeyond eGate Integrator JMS Reference Guide*

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

Enabling Monitoring for eView Studio

You can use the Enterprise Manager to monitor most components of an eView Studio system, including Collaborations, Business Processes, and eWays. In order to view the special tools for each component type, you must have the Enterprise Manager Plug-in files uploaded and installed for those types. For more information about installing the eView Enterprise Manager Plug-in, see **"Installing the eView Enterprise Manager Plug-in" on page 44**. For information on installing the plug-in for eWays, eInsight BPM, and other components, see the appropriate user's guide. Figure 66 shows the **Summary** tab of the Monitor for an eView Studio server Project and Collaboration Project. You can click on any of the displayed components for more information about their status, alerts, log entries, and so on.

Figure 66 Monitoring eView Studio Project Components

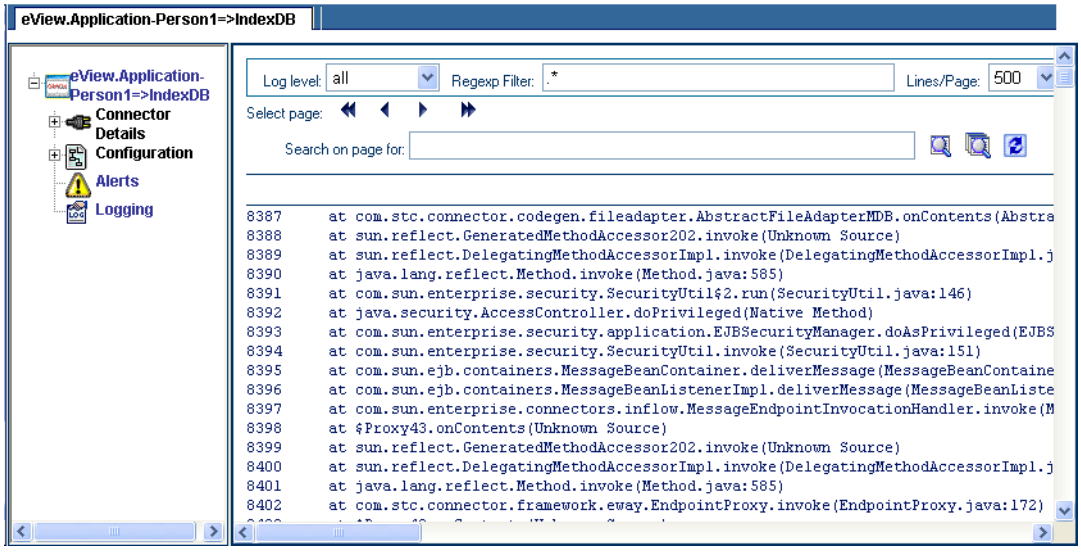


12.3.2 Log Files

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

Figure 67 shows a log file for the Oracle eWay in the eView Studio server Project. For more information on log files, see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

Figure 67 eView Studio Database Logging

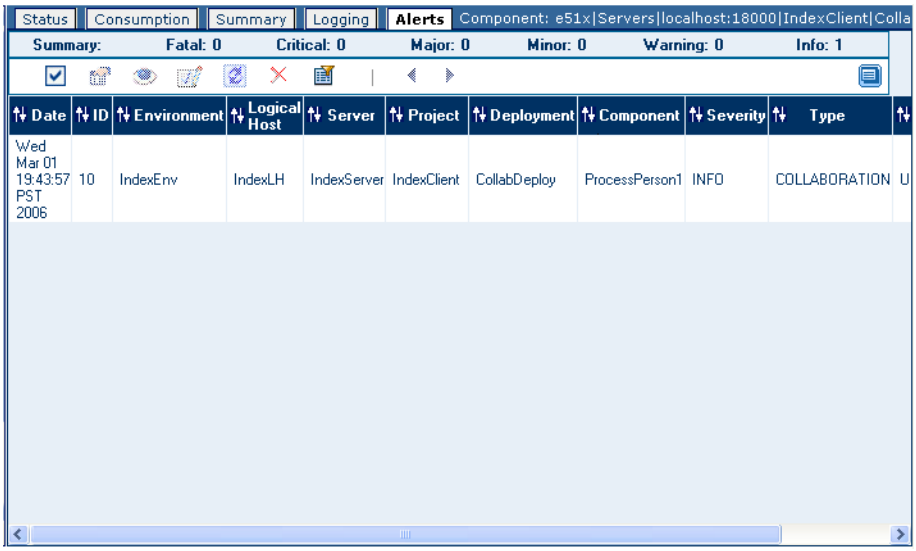


12.3.3 Alerts

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

Figure 68 shows an alert for a Collaboration in an eView Studio client Project. A list of eView Studio alerts appears in Table 14. For more information on alerts, see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

Figure 68 Alert for eView Studio Collaboration



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

Table 14 eView Studio Alert Messages

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

Table 14 eView Studio Alert Messages

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

Table 14 eView Studio Alert Messages

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

Table 14 eView Studio Alert Messages

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

12.4 Implementing Changes to the eView Studio Project

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

12.4.1 Modifying Configuration Files

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

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

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

Updating the Object Structure

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

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

Updating Normalization and Standardization Structures

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

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

Updating the Match String

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

12.4.2 Modifying Standard Project Components

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

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

12.4.3 Modifying the Database

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

12.4.4 Modifying Security

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

12.4.5 Modifying the Local ID Format

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

Chapter 13

Implementing the eView Studio Sample

Sun provides sample Projects for a master company index so you can quickly start working with an eView Studio application and learn about the features. You can install and work with the samples to better understand how the master index works and how various components of eView Studio correlate. This chapter explains how to import the sample files, customize the Projects, and then process data through the eView Studio application using the client Projects and the Enterprise Data Manager (EDM).

What's in This Chapter

13.1 About the Sample Projects

The eView Studio sample implements a simple master company index. It includes one server Project and two client Projects. The sample also provides two small data files in XML format so you can immediately enter data into the master index through a Collaboration or through an eInsight Business Process (BP). Once you implement the master index, you can create your own data files to enter data through the client Projects and you can create and modify data through the EDM. There are three Projects included in the sample.

13.1.1 The Server Project

The eView Studio server Project defines a simple master index application that processes and stores company information using the default Company template as a basis. The sample uses the SBME to standardize data and provide matching weights. The Connectivity Map defines the eView Studio application and also defines the database connection pool using an Oracle eWay, which is configured to connect to the database using a thin client JDBC connection. The server Project includes a method

OTD that contains the methods used in the client Projects' Java Collaboration and Business Process. It also includes an outbound OTD that you can use with a JMS Topic to broadcast messages processed by eView Studio back out to external systems.

The Object Structure

The object structure for the eView Studio application is based on the standard Company template provided by the eView Wizard. It includes information essential to identifying a company, such as its name, stock symbol, tax payer ID, and so on. It also includes address and phone child objects, as well as parsed and phonetic versions of the company name and street address fields.

Query Configuration

Three different types of queries are defined in the Candidate Select file of the eView Studio sample: a basic alphanumeric query, a phonetic version of the alphanumeric query, and a blocking query to use for potential duplicate processing. The blocking query contains the following data blocks, which you can modify.

- The phonetic version of the company name
- The company's stock symbol
- The company's Standard Industrial Classification (SIC)
- The company's tax payer ID
- The house number and phonetic street name of the company's address (this block contains a hint that creates an index against the address tables)
- The number of employees (this is a range supplied by the user or by the incoming message)

Standardization and Matching Configuration

The sample server Project is configured to parse, normalize, and phonetically encode company information and street address information. The following sections provide a general overview of how standardization and matching are configured. You can customize the configuration to obtain different results.

Parsing the Company Name

The eView Studio sample application expects that most company information, such as the name, organization type, industry code, and so on, is included in one freeform text field and must be parsed and normalized. It also expects that street address information must be also parsed, normalized, and phonetically encoded. The company name is parsed into the following components (with the actual field name in parentheses).

- Company name (CompanyName_Name)
- Organization type (CompanyName_OrgType)
- Association type (CompanyName_AssocType)
- Industrial sector (CompanyName_Sector)

- Industry type (CompanyName_Industry)
- Alias (CompanyName_Alias
- Web site address (CompanyName_Url)

After being parsed, the company name is also phonetically encoded.

Parsing Addresses

In the sample, addresses are parsed and standardized based on multiple national domains, including France, Australia, United Kingdom, and United States. The default value, used when no domain is specified, is the United States. Addresses are parsed into the following components (with the actual field name in parentheses).

- House number, rural route identifier, P.O. box number, (AddressLine1_HouseNo)
- Match street name, rural route descriptor, P.O. box descriptor (AddressLine1_StName)
- Street direction (AddressLine1_StDir)
- Street type (AddressLine1_StType)

After being parsed, the street name is also phonetically encoded.

The Match String

The following match string is defined for the eView Studio application (with the actual field name in parentheses). You might want to remove some of the fields that contain less reliable data in terms of matching, such as the association type, street direction, or street type.

- Company name (CompanyName_Name)
- Organization type (CompanyName_OrgType)
- Association type (CompanyName_AssocType)
- Industrial sector (CompanyName_Sector)
- Industry (CompanyName_Industry)
- URL (CompanyName_Url)
- Street name (AddressLine1_StName)
- Street number (AddressLine1_HouseNo)
- Street direction (AddressLine1_StDir)
- Street type (AddressLine1_StType)

Threshold Configuration

Table 15 lists the default configuration for the Threshold file of the sample server Project. The query builder specified for matching is the query named “BLOCKER-SEARCH” in the Candidate Select file.

Table 15 Default Threshold Parameter Configuration

Parameter	Default Configuration
Update Mode	Optimistic
OneExactMatch	False
SameSystemMatch	True
Duplicate Threshold	7.25
Match Threshold	29.0
EUID Length	10
Checksum	No
Chunk Size	1000

Enterprise Data Manager Configuration

In addition to the standard EUID and System/Local ID lookups, three additional searches are defined for the EDM: an alphanumeric search, a phonetic search, and a blocker search. The following criteria fields are defined for these searches.

- CompanyName
- StockSymbol
- TaxPayerID
- CompanyType
- NoOfEmployees
- AddressLine1
- City

In order to perform a search, the CompanyName field must be entered and either the StockSymbol or the CompanyType must also be entered. You can change the fields required for a search in the Enterprise Data Manager file of the Project. The NoOfEmployees field is defined to support range searching and includes a "From" field and a "To" field on the EDM search pages, allowing you to supply the range of values on which to search.

In the default configuration, the Search page is the default to appear when you log on to the EDM, all standard reports are enabled, and the audit log is not enabled. If you want to view the audit log for the sample Project, be sure to enable it before generating the Project.

13.1.2 The Collaboration Client Project

The eView Studio sample provides a Collaboration client Project that shares information between the eView Studio application and two File eWays (one sending data to eView Studio and one receiving data from eView Studio). The client reads the sample data from the input file through the inbound File eWay, unmarshals the data

into the eViewSample_Data OTD, and copies each field value of the OTD into the class variable **varSystemCompany**. The variable is then passed to the Java method **executeMatch**, which processes the new data by comparing it against existing eView Studio records, assigning matching weights between records, and determining whether the incoming data is a new record or an update to an existing record.

After the message is processed, the EUID for the new or updated record is sent to the outbound File eWay. You can call additional eView Studio methods in the Collaboration to customize the processing logic. To view the available methods, right-click "Company_1" in the left pane of the Business Rules Designer and then select **Browse this type**.

13.1.3 The eInsight Client Project

The eView Studio sample provides an eInsight client Project that shares information through an eInsight Business Process. This Project uses the same processing logic as the Collaboration client Project, using **executeMatch** to process messages into the eView Studio application and returning the EUID of the affected record. This Project also uses the same DTD OTD as the Collaboration client Project. You can call additional eView Studio methods in the Business Process to customize the processing logic. Available methods are listed under the **Company** node of the eView Studio server Project.

13.2 Requirements

In order to work with the sample Projects, you must have the following Sun SeeBeyond applications installed.

- eGate Integrator
- Enterprise Designer
- Enterprise Manager
- Oracle eWay
- File eWay
- eInsight BPM (only if you want to work with the eInsight client Project)

You must also have access to a standard Oracle database, either on your computer or over the network. If you do not have eInsight BPM installed, you can still work with the other Projects; simply ignore the instructions for configuring and working with the eInsight client Project since this Project will not install without eInsight BPM. The instructions are based on the assumption that you are running eView Studio on the Sun SeeBeyond Integration Server (IS).

13.3 Importing the Sample Projects

In order to work with the eView Studio sample Projects, you must import the Projects into your Enterprise Designer. Before you begin this step, make sure you have installed the sample files, as described in **"Accessing the eView Studio Documentation and Sample" on page 45**.

To import the sample Projects

1. In Enterprise Designer, right-click the Repository name.
2. From the context menu, select **Import Project**.
3. On the **Import** dialog, click **Yes** to continue or click **No** to save any changes and restart the import process.
4. In the **From ZIP File** field on the **Import Manager** window, browse to the directory where you downloaded the sample files, and then select **eView_Sample.zip**.

   Three Projects appear in the list to import.
5. Click **Import**.
6. On the **Import Status:** dialog, click **OK**.

Note: *Messages might appear warning you of missing components. You can ignore these messages as long as you have the sample requirements (listed in* **Requirements** *on page 175**) installed.*

7. On the **Import Manager** window, click **Close**.

   The eView Studio sample Projects are now visible in the Project Explorer.

13.4 Implementing the Sample Projects

Before continuing, make sure you have access to an Oracle database. The database can be installed on the same computer as eView Studio or on any computer that can be accessed over your network.

Implementing the sample Project includes the following tasks:

13.4.1 Customize the Application

The sample Projects work in their default configuration, but you can modify the configuration files to customize several attributes of the sample eView Studio application, such as the appearance of the EDM, the queries used by the application, the matching parameters, and so on. In order for the sample client Projects to work with the application, however, you cannot modify the object structure. Make any customizations before continuing to the following steps. Here are some recommended changes.

- In the Match Field file, change "OrigStreetName" to "MatchStreetName" in the standardization structure for addresses.
- In the Match Field file, remove some of the extraneous fields from the match string. This will give you a better idea of how matching probability weights are generated. Fields to consider removing include AddressLine1_StType, AddressLine1_StDir, and any of the CompanyName_* fields that are not included in your test data. You can also add any of the other fields in the object structure to the match string.
- In the Enterprise Data Manager file, enable audit logging (change the value of the **allow-insert** element at the end of the file to "true"). If this is not enabled, you will not be able to view the audit log on the EDM.
- In the Enterprise Data Manager file, the Search page is configured such that the CompanyName field is required for a search and either the StockSymbol or CompanyType field is also required. To make searches from the EDM easier, you can change the **required** attribute to false for any of these fields.

13.4.2 Regenerate the Application

Before you can work with the application files, you must regenerate the application and then import the regenerated .jar files into the eView_Sample_Collab_Client Project Collaboration.

Important: *When you regenerate, the database scripts are updated to reflect object structure changes except the Create Company Indexes script, which is not an automatically generated file. If any changes to this script are required after changes to the object structure, you must modify the file manually.*

To regenerate the application

1. In the Project Explorer, expand the **eView_Sample** Project.
2. Right-click **eView Application - Company**.
3. On the context menu, click **Generate**.
4. On the Confirm dialog, click **Yes**.
5. When the application is regenerated, close the output window.

To update the Collaboration

1. In the Project Explorer, expand the eView_Sample_Collab_Client Project.
2. Check out and open the eViewSampleJavaCollab Collaboration.

3 In the toolbar, click **Import JAR File**. The Add/Remove Jar Files dialog appears.

4 For each eView Studio .jar file in the list, highlight the filename and then click **Remove**.

5 For each eView Studio .jar file to re-import, do the following

A Click **Add**.

B Double-click the eView_Sample Project name in the list that appears.

C Select the name of the .jar file to import.

D Click **Import**.

13.4.3 Create the Database Tables

To create the sample database tables, you must have a standard Oracle database installed either on your computer or on a network computer accessible to your machine. To create the database tables, you can use the database scripts in their default form or you can add additional systems and common table data (see **"Step 3: Customize the Database Scripts" on page 99** for more information). Make sure not to delete or change any of the existing information.

To create the database tables

1 If you have not done so already, create a standard Oracle database using Oracle database tools.

2 Create a user for the database. Use the following script as a sample, entering the user name and password of the eView Studio user who will create the database files and provide the connection from the EDM and eWays.

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

3 In the Project Explorer, expand the **eView_Sample** Project, expand **eView Application - Company**, and then expand the **Database Script** folder.

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

5 On the Properties window, do the following.

A Change <host> to the name of the computer on which the database resides (you can enter "localhost" if the database is on your local computer).

B Change <SID> to the SID name of the eView Studio database.

C Enter the user ID for the user you created in step 2 above.

D Enter the password for the user you created in step 2 above.

E Close the dialog.

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

7 Right-click **Create Company Indexes**, and the click **Run**.

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

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

13.4.4 Configure the Connectivity Maps

The Connectivity Maps for all three sample Projects are predefined. You only need to configure the eWay connections for the client Projects. The Oracle eWay is already configured for thin client connectivity in the server Project.

To configure the client connectivity maps

Note: *Do the following for both the Collaboration and Business Process Project.*

1. Check out the Connectivity Map (**CMap1**), and then open it in the Connectivity Map Editor.
2. In the Connectivity Map Editor, double-click the eWay icon between the Service and the eView Studio application.

   The red warning circle disappears.
3. You can define properties for the File eWays, but this is no necessary. To view the default properties, double-click the icon between the eWay and the Service.
4. Save any changes to the Repository.

13.4.5 Define the Environment

In order to deploy the Projects to a Logical Host, you must create an Environment that will run both the server and client Projects.

To define the Environment

1. In the Environment Explorer, right-click the Repository name, and then select **New Environment**. Rename the Environment to "eViewEnvironment".
2. Right-click **eViewEnvironment**, point to **New**, and then select **Logical Host**. Rename it to "eViewLogicalHost".
3. Right-click **eViewLogicalHost**, point to **New**, and then select Sun SeeBeyond Integration Server. Rename it to "eViewServer".
4. Right-click **eViewServer**, select **Properties**, and then define the Integration Server password (this is the Administrator password). Click **OK** to close the Properties window.
5. Right-click **eViewEnvironment**, point to **New**, and then select **File External System**. Enter "FileSystem" for the name.
6. Right-click **FileSystem** and configure the File eWays as follows.
   - For the inbound File eWay, expand **Inbound File eWay**, select **Parameter Settings**, and then specify the directory in which you will place the sample data files from the sample Project.
   - For the outbound File eWay, expand **Outbound File eWay**, select **Parameter Settings**, and then specify the directory to which you want the eWay to write the outbound files.

See the *Sun SeeBeyond File Adapter User's Guide* for more information about configuring File External Systems.

7 Right-click **eViewEnvironment**, point to **New**, and then select **Oracle External System**. Enter "OracleSystem" for the name.

8 Configure **OracleSystem** as follows.

A Right-click **OracleSystem**, and then select **Properties**

B Expand **Outbound Oracle** eWay, and then select **JDBC Connector Settings**.

C Define the properties listed in Table 16.

D Click **OK** to close the Properties window.

Table 16 Oracle External System Properties

Property	Description
ServerName	The name of the database server (you can enter "localhost" if the database is on the same computer as the Logical Host).
DatabaseName	The TNS name of the database.
User	The login ID of the administrator user you created when you created the database (under Create the Database Tables on page 168).
Password	The database password for the administrator user.

See the *Sun SeeBeyond eWay Adapter for Oracle User's Guide* for more information about configuring the Oracle External system.

13.4.6 Create and Start a Domain

Before defining Deployment Profiles for your Projects, make sure you have created and started an instance of the Logical Host containing the integration or application server to which the Project will be deployed. This section describes how to create a domain using the Domain Manager. You can also create the instance using a command-line tool. The command-line method is described in the *Sun SeeBeyond eGate Integrator System Administration Guide*.

Important: *Before you start the domain, be sure that the eView Studio database is running.*

To create a domain using the Domain Manager

1 In the **<jis51>\logicalhost** directory, run the **domainmgr.bat** script.

2 If there are currently no domains, a dialog box indicates that you can create a domain now. If you click **Yes**, the **Create Domain** dialog box appears. Go to step 4.

The Domain Manager appears.

3 On the Domain Manager toolbar, click the **Create a New Domain** tool. The **Create Domain** dialog box appears.

4 If you modified the default port numbers for the server in the Environment, modify the port numbers for the domain accordingly.

5 Click **Create**.

6 When the **Message** dialog box indicates that the domain has been successfully created, click **OK**.

7 (Make sure the eView Studio database is running before performing this step.) On the Domain Manager window, select the new domain and then click **Start an Existing Domain**.

8 When the **Message** dialog box indicates that the domain has been successfully started, click **OK**.

13.4.7 Deploy the Projects

For each sample Project, you must create a Deployment Profile and build and deploy the Project. Be sure to build the server Project before creating the client Projects' Deployment Profiles. Building the server Project creates the eView Studio application, which is required in order to deploy the client Projects.

Deploy the Server Project

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

To deploy the server Project

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".

4 Select **eViewEnvironment** for the Environment.

5 Select only **CMAP1** as the Connectivity Map to use.

6 Click **OK**.

The Deployment Editor appears.

7 In the Deployment Editor window, click the **Automap** icon.

The Project components are automatically mapped to the Environment containers (the eView Studio application components are mapped to the server and the Oracle eWay is mapped to the Oracle external system).

8 In the **Deployment Editor** toolbar, click **Build**.

The application .ear file is generated and placed in **<edesigner>\builds\<project_name><deploy_profile_name>\<logicalhost_name>\<server_name>**.

9 After the deployment builds successfully, click **Deploy** in the **Deployment Editor** toolbar.

The application .ear file is extracted to the domain.

Deploy the Collaboration Client Project

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

To deploy the Collaboration client Project

1. In the Project Explorer, right-click the **eView_Sample_Collab_client** Project.
2. Point to **New**, and then click **Deployment Profile**.
3. Name the profile "CollabDeployment".
4. Select **eViewEnvironment** for the Environment.
5. Select only **CMAP1** as the Connectivity Map to use.
6. Click **OK**.

   The Deployment Editor appears.
7. In the Deployment Editor window, click the **Automap** icon.

   The Project components are automatically mapped to the Environment containers.
8. In the **Deployment Editor** toolbar, click **Build**.

   The application .ear file is generated and placed in **<edesigner>\builds\<project_name><deploy_profile_name>\<logicalhost_name>\<server_name>**.
9. After the deployment builds successfully, click **Deploy** in the **Deployment Editor** toolbar.

   The application .ear file is extracted to the domain.

Deploy the eInsight Client Project

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

To deploy the eInsight client Project

1. In the Project Explorer, right-click the **eView_Sample_Collab_client** Project.
2. Point to **New**, and then click **Deployment Profile**.
3. Name the profile "eInsightDeployment".
4. Select **eViewEnvironment** for the Environment.
5. Select only **CMAP1** as the Connectivity Map to use.
6. Click **OK**.

   The Deployment Editor appears.
7. In the Deployment Editor window, click the **Automap** icon.

   The Project components are automatically mapped to the Environment containers.
8. In the **Deployment Editor** toolbar, click **Build**.

   The application .ear file is generated and placed in **<edesigner>\builds\<project_name><deploy_profile_name>\<logicalhost_name>\<server_name>**.

9 After the deployment builds successfully, click **Deploy** in the **Deployment Editor** toolbar.

The application .ear file is extracted to the domain.

13.4.8 Define EDM Security

To run the eView Studio application, you must connect to the server using the Enterprise Manager and then create a user for logging on to the Enterprise Data Manager (EDM). For more information about working with the Enterprise Manager, see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

Connect to the Server

Before you can create an EDM user account, you must connect to the server from the Enterprise Manager.

To connect to the server

1 Log in to the Enterprise Manager.

2 In the frame on the right side of the page, click **J2EE**.

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

3 In the **Add Application Server** section, fill in the fields listed in Table 17.

Table 17 Manager Server Fields

In this field …	Type or select …
Server Type	The type of server you are adding (either Sun SeeBeyond Integration Server or Sun Java System Application Server).
Host Name	The name of the computer on which the server resides.
HTTP Administration Port	The HTTP port for connecting to the server. By default, the port number is **18000**.
Username	The user name to log on to the application or integration server.
Password	The password associated with the given user name.

4 Click **Connect to Server**.

5 Once the connection is made, click **Save current user preferences**.

Create a User Account

Following the instructions under **"Defining Security" on page 155**, create a new user logon ID and password (for logging on to the EDM). Add the user to the **eView.Admin** group.

To create a user account

1. On the Enterprise Manager, right-click the server you added in **"Connect to the Server" on page 183**, and then click **Manage Integration Server Users**.

   The **Users List** window appears.

2. In the **Users List** window, click **Add New User**.

   The **Add/Edit User** window appears.

3. In the **User Name** field, enter "eview".
4. In the **Password** field, enter "eview".
5. In the **Confirm Password** field, enter "eview" again.
6. In the **Group List** field, enter "eView.Admin".
7. You can enter different user groups to see the affect each has on permissions. Each user must be assigned to either eView.Admin or eView.User. **Table 13 on page 157** lists and describes each eView Studio user group.
8. After you have added all required user groups, click **Submit**.

Redeployment Notes

You can disable, enable, deploy, and undeploy applications through the Enterprise Manager without using Enterprise Designer. The .ear files for the sample Projects are located in the following directories. See the *Sun SeeBeyond eGate Integration System Administration Guide* for information on performing any of the deployment tasks from the Enterprise Manager.

- edesigner\builds\eView_SampleeViewDeployment\eViewLogicalHost\eViewServer\eView_SampleCompanyDeploy.ear
- edesigner\builds\eView_Sample_Collab_clientCollabDeployment\eViewLogicalHost\eViewServer\eView_Sample_Collab_clientCollabDeployment.ear (to run data through the Collaboration)
- edesigner\builds\eView_Sample_BP_ClienteInsightDeployment\eViewLogicalHost\eViewServer\eView_Sample_BP_ClienteInsightDeployment.ear (to run data through the Business Process)

13.5 Working With the eView Studio Sample

You can work with the eView Studio sample application by entering information through the EDM and by sending information to the master index through the Collaboration or Business Process using the provided sample files. You can also use these files as a template to create your own data to enter into the system.

Run the Sample Files

Running the sample files inserts company records into the eView Studio sample database. The data in both sample files is identical, so if you run both files only one record should be created depending on how the match string and match thresholds have been modified. To access the sample files, navigate to the directory where the **eView_Sample.zip** file is located and extract the following two text files to the directory you specified for the inbound File eWay in **"Define the Environment" on page 179**.

- eViewSampleBP_input.txt
- eViewSampleCollab_input.txt

The inbound File eWays pick up the files and append ".~in" to the file name so you know that each was picked up. After the data is processed, the outbound File eWays create an output file for each input file in the directory you specified when you defined the Environment. The files contain the output of **executeMatch**.

Note: *Both files contain the same data. If you process them both through eView Studio, only one record might appear in the database (depending on whether you modified the match string and thresholds) because the master index will match the two records.*

Work with the EDM

Using the EDM, you can view the records created by processing the sample data files. You can also create new records, compare records, merge records, and so on. For instructions on working with the EDM, see the *Sun SeeBeyond Enterprise Data Manager User's Guide*.

To log on to the EDM

1. Open a web browser.
2. In the Address field, type the following:

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

where <*port*> is the HTTP port number, which appears on the Domain Manager in the HTTP field (by default, the port number is 18001).

To view the new record

1. When you log on, the first page to appear is the Search page.
2. If you entered data through the File eWays do the following.
   A. In the Company Name field, enter "Intel".
   B. In the Stock Symbol field, enter "INTC"
   C. Click **Search**.

      The Search Results page appears.

   D. To view detailed information about the resulting record, click the EUID of the record.

3 To add a new record, click **Create System Record**.

Appendix A

Field Notations

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

What's in This Appendix

A.1 ePath

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

ePath strings can be of four basic types:

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

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

```
Person.Alias[@type=Main]
```

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

```
Person.Alias[lastname=Jones]
```

- **Field** - designates the field to return, and is in the form of a string.

Example

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

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

```
PostalCode = 90245
```

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

A.2 Qualified Field Names

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

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

A.3 Simple Field Names

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

Syntax

The syntax of a simple field name is:

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

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

Initial Load Tips

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

What's in This Chapter

B.1 Legacy Data

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

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

B.2 Database Indexes

The database indexes required for the initial load process differ somewhat from the optimal indexes for running eView Studio in production. At a minimum, you should

create an index for each block defined in the blocking query used for matching. It will also help to remove the indexes from the sbyn_transaction and sbyn_potentialduplicate tables. After the initial load is complete, you can restore any deleted indexes. The code for restoring the indexes can be found in the database creation script in the eView Studio server Project.

B.3 Threshold Parameters

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

B.4 Reports

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

Appendix C

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.

What's in This Appendix

C.1 About Match and Standardization Types

For each field that will be used for matching in the new index, you can select a *match type* in the eView Wizard. When you select a match type for a field, eView Studio automatically adds that field to the match string in the Match Field file and, in many cases, generates additional fields in the Object Definition that are not visible on the eView Wizard. These fields are used for searching and matching and they 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 store 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 might also be defined for standardization in the Match Field file, in which case they will also be assigned a standardization type. The types described in this appendix pertain to the Sun SeeBeyond Match Engine only.

Note: *The match types specified in the Match Field file for the fields in the match string are not always 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* ***Implementing the Sun SeeBeyond Match Engine with eView Studio****.*

C.2 Sun SeeBeyond Match Engine

The eView Wizard match types for the Sun SeeBeyond Match Engine (SBME) fall into four primary categories.

- **Person Match Types** on page 195
- **BusinessName** on page 196
- **Address** on page 196
- **Miscellaneous Match Types** on page 197

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**.

C.2.1 Person Match Types

The Person match types include PersonLastName and PersonFirstName. 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 specify a field with a person match type for blocking in the eView Wizard, the phonetic version of the name is automatically added to the blocking query. The following fields are created when you specify one of the Person match types for a field (<*field_name*> refers to the name of the field specified 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 and match types in the Match Field file are listed in Table 18.

Table 18 Person Name Standardization and Match Types

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

C.2.2 BusinessName

The BusinessName match type is designed to help parse, normalize, and phonetically encode a business name. BusinessName matching adds several fields to the Object Definition and to the match string. If you specify 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 all 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.

Table 19 lists the fields created when you select the BusinessName match type for a field 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.*

Table 19 BusinessName Match Types

Field Name	Description	Added to the Match String?	Match Field Match Type
<field_name>_Name	The parsed and normalized version of the business name.	Yes	PrimaryName
<field_name>_NamePhon	The phonetic version of the business name.	No	
<field_name>_OrgType	The parsed organization type of the business name.	Yes	OrgTypeKeyword
<field_name>_AssocType	The association type for the business.	Yes	AssocTypeKeyword
<field_name>_Industry	The name of the industry for the business.	Yes	IndustryTypeKeyword
<field_name>_Sector	The name of the industry sector (industries are a subset of sectors).	Yes	IndustrySectorList
<field_name>_Alias	An alias for the business name.	No	
<field_name>_Url	The business' web site URL.	Yes	Url

C.2.3 Address

The Address match type is designed to help 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 specify 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.*

Table 20 Address Match Types

Field Name	Description	Added to Match String?	Match Field Match Type
<field_name>_HouseNo	The parsed street number of the address.	Yes	HouseNumber
<field_name>_StDir	The parsed and normalized street direction of the address.	Yes	StreetDir
<field_name>_StName	The parsed and normalized street name of the address.	Yes	StreetName
<field_name>_StPhon	The phonetic version of the street name.	No	
<field_name>_StType	The parsed and normalized street type of the address, such as Boulevard, Street, Drive, and so on.	Yes	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.

C.2.4 Miscellaneous Match Types

Several additional eView Wizard match types are defined for the SBME. 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 contains 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 specify one of these match types for a field in the eView

Wizard, the field is added to the match string with a match type of **String**, **Date**, **Number**, or **Char**.

Glossary

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

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

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

Candidate Select file
The eView Studio 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 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. See also *matching threshold*.

EDM
See *Enterprise Data Manager*.

Enterprise Data Manager
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. Also known as the EDM.

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

LID
See *local ID*.

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. The combination of a local ID and system constitutes a unique identifier for a system record. The name of the local ID field is configurable on the EDM, and might have been modified for your implementation.

master index
A database application that centralizes and cross-references information on specific objects in a business organization.

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

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

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

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

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

Matching Service
An eView Studio component that defines the matching process. This component is configured by the Match Field file.

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

matching weight *or* match weight
See *matching probability weight*.

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

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

system
A computer application within an organization where information is entered about objects and that shares information with the master index (such as a registration system). Also known as a source system, local system, or external system.

system object
A record received from a local system. The fields contained in system objects are used in combination to populate the SBR. The system objects for one 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 Create System Record tab to display the Create System Record page.

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

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

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

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X