SeeBeyond ICAN Suite

eView Studio Configuration Guide

Release 5.0.4

SEEBEYOND®

Contents

List of Tables

Chapter 1

Introduction

This guide provides comprehensive information on customizing the configuration files for a SeeBeyond® eView Studio (eView) enterprise-wide master index. As a component of SeeBeyond's Integrated Composite Application Network (ICAN) Suite, eView helps you integrate information from disparate systems throughout your organization. This guide explains how to configure processing components of a master index, including the object structure, runtime environment, and Enterprise Data Manager (EDM). The configuration files described in this guide appear in the **Configuration** folder of the eView Project and also include the Object Definition file.

This guide is intended to be used with the *eView Studio User's Guide* and *Implementing the SeeBeyond Match Engine with eView Studio* or *Implementing Ascential INTEGRITY with eView Studio*, depending on the match engine you choose.

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

1.1 Document Purpose and Scope

This guide provides step-by-step instructions for configuring certain processing components of a master index. It includes navigational information, functional instructions, and background information where required. A summary of configuration activities is provided in **Chapter 2** of this guide.

This guide does not include information or instructions on creating a master index, configuring the database or connectivity components, or using the Enterprise Data Manager (EDM). These topics are covered in the appropriate user guide (for more information, see **"Supporting Documents" on page 15**).

1.1.1 Intended Audience

Any user who configures processing properties for an eView master index should read this guide. A thorough knowledge of eView is not needed to understand this guide, but familiarity with the application is helpful. It is presumed that the reader of this guide is familiar with the eGate environment and GUIs, the operating system(s) on which eGate and the index database run, and web services. The reader should also be familiar with XML documents.

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

1.1.2 Using this Guide

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

1.1.3 Document Organization

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

- Chapter 1 **"Introduction"** gives a general preview of this document—its purpose, scope, and organization—and provides sources of additional information.
- Chapter 2 **"Configuration Overview"**gives overview information about the configuration files created by the eView Wizard.
- Chapter 4 **"Object Definition"** describes the Object Definition file and how to configure the definition of the primary object to be stored in the master index.
- Chapter 3 **"Candidate Select Configuration"**describes how to define and customize the queries that are used by the Enterprise Data Manager (EDM) to search the database and by the master index to find a subset of records that are potential matches of an incoming record.
- Chapter 5 **"Threshold Configuration"** describes how to customize certain properties of the match process, such as setting thresholds, specifying a blocking query, defining EUID properties, and so on.
- Chapter 6 **"Match Field Configuration"** describes how to configure the match and standardization engines, including defining the match string, specifying fields for standardization and phonetic conversion, and specifying Java classes for certain matching functions.
- Chapter 7 **"Best Record Configuration"** describes how to configure the survivor calculator to define the logic for creating the single best record (SBR). It also describes the classes used to control update procedures.
- Chapter 8 **"Field Validation Configuration"** describes how to configure the field validator and provides information about creating custom field validators.
- Chapter 9**"Enterprise Data Manager Configuration"**describes how to configure the EDM and to set up each client workstation to be able to access the master index database using the EDM.

1.2 Writing Conventions

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

1.2.1 Special Notation Conventions

The following special notation conventions are used in this document.

Table 1 Special Notation Conventions

Text	Convention	Example
Titles of publications	Title caps in *italic* font	*eView Studio User's Guide*
Button, Icon, Command, Function, and Menu Names	**Bold** text	▪ Click **OK** to save and close. ▪ From the **File** menu, select **Exit**.
Parameter, Variable, and Method Names	**Bold** text	▪ Use the **executeMatch()** method. ▪ Enter the **field-type** value.
Command Line Code and Code Samples	`Courier` font (variables are shown in ***bold italic***)	▪ `bootstrap -p` ***`password`*** ▪ `<tag>Person</tag>`
Hypertext Links	**Blue** text	For more information, see **"Writing Conventions" on page 14**.
File Names and Paths	**Bold** text	To install eView, upload the **eView.sar** file.
Notes	***Bold Italic*** text	***Note:*** *If a toolbar button is dimmed, you cannot use it with the selected component.*

Additional Conventions

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

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

1.2.2 Mouse Conventions

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

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

- **Point** means to position the mouse pointer until the tip of the pointer rests on whatever you want to point to on the screen.

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

1.3 Supporting Documents

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

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

1.4 Online Documents

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

http://www.adobe.com

1.5 SeeBeyond Web Site

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

http://www.SeeBeyond.com

Chapter 2

Configuration Overview

This chapter provides an overview of the configurable components of eView and of the configuration files that define processing properties and the data structure of an eView master index. It also describes the relationships between these files.

2.1 eView Configuration

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

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

2.2 About the Configurable Components

Several components of the master index are configured by the eView Project configuration files. This section lists and briefly describes those components.

2.2.1 Matching Service

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

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

2.2.2 eView Manager Service

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

2.2.3 Query Builder

The Query Builder defines all queries available to the master index. This includes the queries performed automatically by the master index when searching for possible matches to an incoming record. It also includes the queries performed manually through the Enterprise Data Manager (EDM). The EDM queries can be either alphanumeric or phonetic, and have the option of using wildcard characters. The configuration of the Query Builder is defined in the *Candidate Select* file.

2.2.4 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.

2.2.5 Update Manager

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

2.2.6 Database and Object Type Definition

The object structure in the *Object Definition* file defines the structure of the database and of certain dynamic methods in the method Object Type Definition (OTD). Generating an eView Project creates a database script that creates tables and columns into the database based on the object structure. Generating the Project also creates a method OTD that includes the methods you need to use in eView Collaborations or in eInsight to process data into and out of the database.

2.2.7 Enterprise Data Manager

The Enterprise Data Manager (EDM) is a web-based interface that allows you to monitor and maintain the data in the master index database. Many of the configurable properties of the EDM are defined by information you specify in the eView Wizard, but you can further configure the EDM by modifying the Enterprise Data Manager file. The EDM provides the ability to manually search for records; update, add, deactivate, and reactivate records; merge and unmerge records; view potential duplicates and comparisons of object records; view transaction histories; and view audit logs.

2.3 About the Configuration Files

Several XML configuration files define primary characteristics of the master index , such as how data is processed, queried, and matched. These files configure runtime components of the master index , which are listed in **"About the Configurable Components" on page 17**.

Object Definition

In the eView Wizard, you define the objects and fields contained in the object structure, along with properties for those fields. The information you specify is written to the Object Definition file in the eView Project. This file defines the objects stored 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 you define in the Object Definition file determines the structure of the database tables that store object data, the structure of the Java API, and the structure of the OTD generated for the Project.

Candidate Select

This file configures the *Query Builder* component of the master index and defines the available queries. 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. 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.

Match Field

This file configures the *Matching Service* by defining which fields will be used for standardization and matching in the master index. It also specifies which match and standardization engines to use, the query process for matching, and the nationality of the data being standardized. Standardization includes defining fields to be reformatted (parsed), normalized, or converted to their phonetic version. You must also define the data string to be passed to the match engine. The rules you define for standardization and matching are highly dependent on the standardization and match engines in use.

Threshold

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

The Threshold file also configures the EUIDs assigned by the master index . You can specify an EUID length, whether a checksum value is used for additional verification, and a “chunk size”. 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.

Best Record

This file defines the logic for calculating the data to be included in each entity’s SBR. This file allows you to define formulas for determining the data that should be considered the most reliable and how updates to the SBR are handled. The survivor calculator uses these formulas to generate the SBR for a given object. This logic is defined in the **SurvivorHelperConfig** and **WeightedCalculator** sections of the Best Record file.

The SBR is defined by a mapping of fields from external system records. Since there may be many external systems, you can optionally specify a strategy to select the SBR field from the list of external values. You can specify any additional fields that might be required by the survivor strategy to determine which external system contains the best data, such as the record’s update date and time. The flexible structure of this configuration allows you to define custom update procedures as well.

Field Validation

By default, the Field Validation file defines certain validations for the local identifiers assigned by each external system. You can create custom Java classes that define rules for validating field values before they are saved to the master index database. You can then specify the Java classes in the Field Validation file to make them part of the eView application.

Security

This file is not currently used, and is a placeholder to be used in future versions.

Enterprise Data Manager

This file configures the appearance and certain processing properties of the *Enterprise Data Manager* (EDM). In this file, you define each object and 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.

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

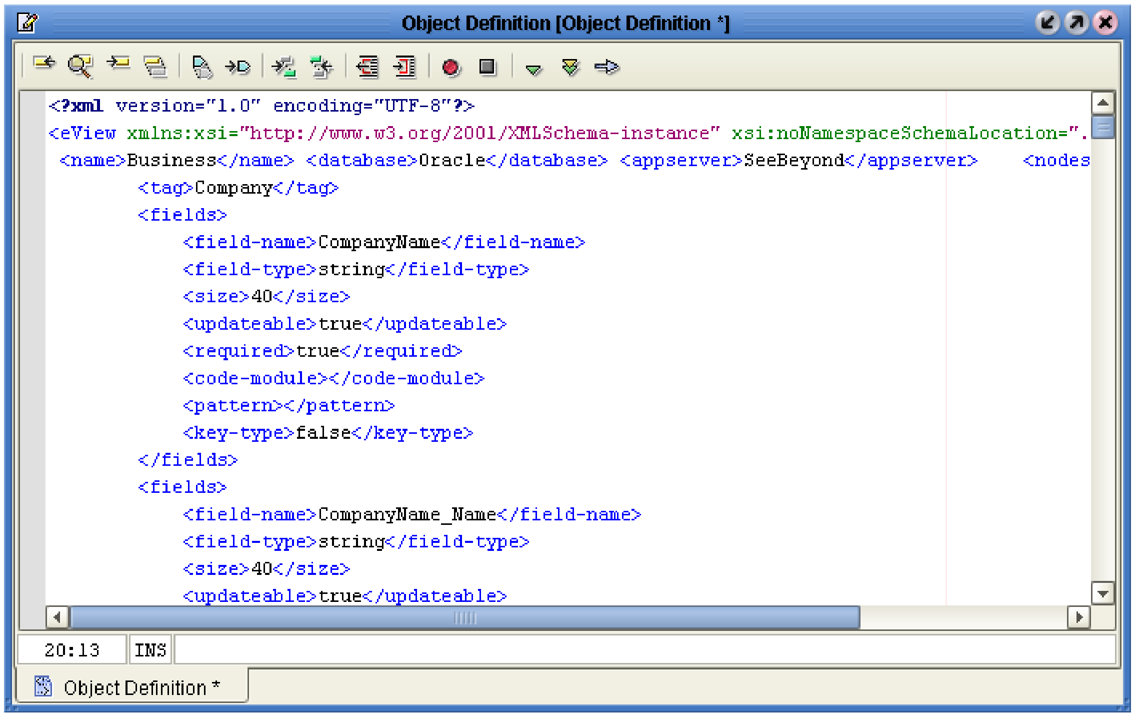
Finally, the Enterprise Data Manager file defines certain implementation information, such as the application or integration server in use, debugging rules, and security activation.

2.4 Using the eView Editors

You can use the XML editor in the Enterprise Designer to modify the configuration files created by the eView Wizard. This editor can only be accessed by selecting a configuration file to open. The XML editor provides several tools to help you format the XML file, check your XML syntax, and search for words or phrases. The editor includes a validation function that is not used (for more information about validations, see **"Modifying Configuration Files" on page 24**).

Figure 1 illustrates the XML editor.

Figure 1 Enterprise Designer XML Editor



```
<?xml version="1.0" encoding="UTF-8"?>
<eView xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation=".
 <name>Business</name> <database>Oracle</database> <appserver>SeeBeyond</appserver>    <nodes
        <tag>Company</tag>
        <fields>
            <field-name>CompanyName</field-name>
            <field-type>string</field-type>
            <size>40</size>
            <updateable>true</updateable>
            <required>true</required>
            <code-module></code-module>
            <pattern></pattern>
            <key-type>false</key-type>
        </fields>
        <fields>
            <field-name>CompanyName_Name</field-name>
            <field-type>string</field-type>
            <size>40</size>
            <updateable>true</updateable>
```

The eView text editor, used to modify the database scripts and SeeBeyond Match Engine configuration files, is similar to the XML editor, but does not include the commands for checking or validating the text. The Java source editor does not include toolbar icons.

2.4.1 XML Editor Toolbar Buttons

The toolbars on the XML and text editors provide one-click shortcuts for executing commands in the editors. Place the cursor over a toolbar button to display the title of that button. Table 2 lists and describes each toolbar button for the XML and text editors.

Table 2 XML Editor Toolbar Buttons

Button	Command	Function
	Find Previous Occurrence	Finds the previous instance of the selected text in the file.
	Find Selection	Highlights all instances of the selected text in the file, or finds the next instance of the selected text, depending on the status of the **Toggle Search Highlight** tool.
	Find Next Occurrence	Finds the following instance of the selected text in the file.
	Toggle Highlight Search	Toggles the **Find Selection** tool to either highlight all instances of the selected text or to find the next instance of the selected text.

Table 2 XML Editor Toolbar Buttons

Button	Command	Function
	Toggle Bookmark	Creates a bookmark in the line in which the cursor is positioned, if none exists. If a bookmark exists in this line, clicking this tool removes the bookmark.
	Next Bookmark	Moves the cursor to the next bookmark in the file.
	Next Matching Word	Changes the value of the selected text to the next word found in the file.
	Previous Matching Word	Changes the value of the selected text to the previous word found in the file.
	Shift Line Left	Shifts the line in which the cursor is positioned to the left.
	Shift Line Right	Shifts the line in which the cursor is positioned to the right.
●	Start Macro Recording	Starts recording the following commands and text entries in a macro.
■	Stop Macro Recording	Stops recording command and text entries after a macro is recorded.
▽	Check XML	Checks the XML syntax of the file. This icon is only available in the XML editor.
	Validate XML	Validates the XML file structure against the schema definition file. This icon is only available in the XML editor.
⇨	XSL Transformation	Converts the active XML file to HTML format. This icon is only available in the XML editor.

2.4.2 Context Menu Commands

XML Editor

If you right-click in the main window of the XML editor, a context menu appears with additional commands

Table 3 XML Editor Context Menu Commands

Command	Function
Open	Opens the XML file in a tree view.
View	Opens the XML file in a web browser.
Check XML	Checks the XML syntax of the file.

Table 3 XML Editor Context Menu Commands

Command	Function
Validate XML	This option is currently inactive.
Generate DTD	This option is currently inactive.
XSL Transformation	This option is currently inactive.
Save	Saves changes made to the active XML file.
Clone View	Opens the active file in a new XML editor window.
Close	Closes the active file.
Open	Opens the XML file structure in a tree-view format.
Edit	This option is currently inactive.
Cut	Removes the selected text and places it on the clipboard.
Copy	Copies the selected text to the clipboard.
Paste	Pastes text that has been cut or copied to the location of the cursor.
Delete	This option is currently inactive.

Text Editor

If you right-click in the main window of the text editor, a context menu appears with additional commands

Table 4 Text Editor Context Menu Commands

Command	Function
Save	Saves changes made to the active text file.
Clone View	Opens the active file in a new, floating text editor window.
Close	Opens the XML file in a tree view.
Open	Opens the active file in a new text editor window.
Edit	This option is currently inactive.
Cut	Removes the selected text and places it on the clipboard.
Copy	Copies the selected text to the clipboard.
Paste	Pastes text that has been cut or copied to the location of the cursor.
Close	Closes the active file.
Delete	This option is currently inactive.

2.5 Modifying Configuration Files

When you modify the configuration files for eView, you must check out the file you are modifying before making any changes. Once you complete the modifications, you should validate the file against its corresponding XML schema definition file to be sure that no eView constraints were violated. For more information about version control and checking files in and out, see the *eGate Integrator User's Guide*.

This section describes the following procedures:

- **Checking out Files** on page 25
- **Validating against the XSD Files** on page 25
- **Saving a File to the Repository** on page 25
- **Checking in Files** on page 26

2.5.1 Checking out Files

Before modifying a file, be sure to check the file out of the Repository.

To check out a file

1. In the Project Explorer, expand the Project tree until you see the configuration file you want to modify.
2. Right-click that file, and click **Check Out**.
3. On the dialog that appears, click **Check Out**.

2.5.2 Saving a File to the Repository

Before modifying a file, be sure to check the file out of the Repository. You can perform this step before or after opening the file.

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 and check it back in to the Repository (described in the following procedures.

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.*

2.5.3 Validating against the XSD Files

eView 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 configuration files

1. After you save change to a 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.

2.5.4 Checking in Files

After you modify and validate a configuration file, check it back into the Repository to make it available to other developers.

To check in a file

1. In the Project Explorer, right click the component you want to check in.
2. On the context menu that appears, click **Check In**.

   The **Version Control - Check In** dialog appears.
3. Enter any descriptive text for the version you are checking in, and then click **Check In**.

Chapter 3

Candidate Select Configuration

You can define the characteristics of the searches performed from the Enterprise Data Manager, as well as the queries used by the master index to search for a candidate pool of potential matches for incoming records. This chapter provides information about queries, the structure of the Candidate Select file, and instructions for modifying the file.

3.1 About Candidate Select Configuration

The Candidate Select file configures properties of the Query Builder, which is a class that uses defined criteria and options to generate queries and query results. The Candidate Select file defines the criteria and options used by the Query Builder. The criteria must be fields defined in the Object Definition, and the options are key and value pairs that fine-tune the query operation. The Candidate Select file defines the properties of queries originating from the EDM as well as queries made for matching purposes.

3.2 Query Builder Components

The Query Builder defines the queries that can be used for searching from the Enterprise Data Manager and the queries the master index uses to search for possible matches to incoming or updated records.

The master index performs two types of queries. Users perform manual queries from the EDM and the index automatically performs queries before processing matches for an incoming record. Two types of queries, *basic queries* and *blocking queries*, are predefined in the Query Builder. In the default installation, basic queries are performed from the EDM and blocking queries are performed for match processing, though this is not required. You can also use a blocking query for the phonetic searches performed from the EDM. Both types of queries are configured by the Candidate Select file, and custom queries can be created and implemented with the master index .

You can configure certain query properties. Both types of queries can be configured to search on standardized or phonetic versions of the search criteria. Basic queries can be configured to allow wildcard characters. For the blocking queries, you define the criteria to include in each block of query criteria.

3.2.1 Basic Queries

By default, searches performed from the EDM follow the logic defined in the configured basic queries. You can specify the query to use for each type of search in the EDM (this is defined in the Enterprise Data Manager file). These searches can be weighted, which means that the match engine calculates the likelihood that the search results match the actual search criteria and assigns a matching weight to each returned record. You can also specify whether the search is performed on the phonetic version of the criteria.

The basic query uses all supplied search criteria to create a single SQL query. For this query, each field in the “where” clause is joined by an “and” operator, meaning that only records that match on all search fields are returned. This query has an option to allow wildcard characters in the search criteria. When this option is set to **true**, the query uses the “like” database operator rather than “equal”. This option allows you to search on criteria for which you have incomplete data.

The searches performed from the EDM can be further customized by modifying the Enterprise Data Manager file (for more information, see **“Configuring the Search Pages” on page 149**).

3.2.2 Blocking Queries

When the master index evaluates possible matches of records sent to the index from the external systems and from the EDM, the index performs a set of predefined SQL queries to retrieve a subset of possible matches. These queries are known as *blocking queries*. The matching algorithm processes the input record against the profiles retrieved from the blocking query (known as the *candidate pool*) and assigns them matching probability weights.

The Candidate Select file allows you to define the criteria and conditions for querying the database to retrieve the subset of possible matches to the incoming record. You can define multiple queries, known as “blocks”, for each blocking query, and the master index performs each of these queries in turn when processing records until sufficient records are retrieved (called a “match pass”). Each field in a block is joined by an “and” operator in the “where” clause, and each block is joined by an “or” clause. This type of search is also useful as a phonetic search in the EDM.

The blocking queries you define here are referenced in the Threshold file, which specifies which one of the defined blocking queries to use for match processing. They may also be referenced in the Enterprise Data Manager file if a blocking query is used for phonetic searches from the EDM. To enable extensive searching (that is, searching against additional tables, such as an alias table for a person index), you must add the alias fields to the blocking query.

3.2.3 Phonetic Queries

You can configure either the basic query or the blocking query to perform phonetic searches from the EDM. If you use a basic query, then all entered criteria must match existing records in order to return results from the search. If you use a blocking query, several queries are performed using different combinations of data until enough matching records are returned or until all defined combinations have been tried.

For example, if you use a basic query and enter first and last name, date of birth, gender, and SSN for criteria, the basic query might not return any matches if any one of those fields does not match the criteria. However, if you use a blocking query for the same example, it may search on SSN, then on first name and date of birth, and then on last name and gender. The query returns any matching records from all of the query passes.

3.3 The Candidate Select File

The properties for the predefined queries are defined in the Candidate Select file in XML format. Some of the information entered into the default configuration file is based on the fields you specified for blocking in the eView Wizard, and some is standard across all implementations. For most implementations, this file will require customization.

3.3.1 Modifying the Candidate Select File

You can modify the Candidate Select file at any time, but you must regenerate the Project and reactivate the deployment after making any changes to the file. You can either apply the changes immediately to the Logical Host, or restart the host to pick up the changes. The properties of the blocking query used by the match process should not be modified after moving into production because it can cause unexpected matching weight results.

Before making any changes to this file, make sure you are familiar with the procedures described under **"Modifying Configuration Files" on page 24**. The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes.

3.3.2 Candidate Select File Structure

The Candidate Select file is written in XML and is automatically generated by the eView Wizard. This file can be modified using the Enterprise Designer XML editor, which is a standard NetBeans editor. This section describes the structure of the XML file and general requirements and constraints. There are two primary components of the Candidate Select file. The first component, XML File Information, is standard across all implementations. The second component, **QueryBuilderConfig**, defines all queries used in the master index.

XML File Information

The first lines in the Candidate Select file represent information about the schema instance and namespace. You should not modify these components.

Standard Query Properties

Both basic and blocking queries are defined by the same element and properties, but each have different configuration elements. The **query-builder** element identifies each query, and contains several attributes defining the queries. Below is a sample of the standard **query-builder** element attributes.

```
<query-builder name="PHONETIC-SEARCH"
class="com.stc.eindex.querybuilder.BasicQueryBuilder"
parser-class="com.stc.eindex.configurator.impl.querybuilder.
KeyValueConfiguration"
standardize="true"
phoneticize="true">
</query-builder>
```

Table 5 lists and describes the attributes that configure the queries you defined.

Table 5 Query Builder Attributes

Attribute	Description
name	A unique ID for the element. This element is used to identify the Query Builder to use. No spaces are allowed in this attribute.
class	The fully qualified name of the query class. The default Query Builder classes are described in Table 6.
parser-class	The fully qualified name of the class to parse the **config** elements for each query. This should not be modified for the default queries.
standardize	An indicator of whether the query criteria is standardized before being passed to the query. A value of **true** indicates that fields that can be standardized for the query will be; **false** specifies that no fields are standardized for the query.
phoneticize	An indicator of whether the query criteria is phonetically encoded before being passed to the query. A value of **true** indicates that fields can be phonetically encoded for the query will be; **false** specifies no fields are phonetically encoded for the query.

Table 6 lists and describes the default Query Builder classes provided with eView.

Table 6 Default Query Builder Classes

class-name	Description
com.stc.eindex.querybuilder.BasicQueryBuilder	Builds dynamic queries using all the available input fields. When configured to use normalized and phonetic data, this query performs phonetic searches; when configured not to use normalized and phonetic data, this query is used for exact alphanumeric searches.
com.stc.eindex.querybuilder.BlockerQueryBuilder	Builds queries using the criteria defined in the block definitions defined for the query.

Basic Query Configuration

Basic queries are designed to use the key and value options to help fine-tune the query process. Currently, the only key available is the **UseWildCard** key. You can set this to true or false. For example:

```
<config>
    <option key="UseWildcard" value="false"/>
</config>
```

The **config** tags fall within the **query-builder** element.

Blocking Query Configuration

Blocking queries do not use the key and value options. Instead, the **config** elements for blocking queries contain a list of data blocks used for match passes against the database. Each block definition is specified by a **block-definition** element, and identified by a **number** attribute. Within each **block-definition** element, there is one **block-rule** element that contains a list of fields to use as criteria for that **block-definition**. Each field is specified by an **equals** element, and defined by **field** and **source** elements. For example:

```
<config>
    <block-definition number="ID000001">
        <block-rule>
            <equals>
                <field>Enterprise.SystemSBR.Person.SSN</field>
                <source>Person.SSN</source>
            </equals>
        </block-rule>
    </block-definition>
    <block-definition number="ID000002">
        <block-rule>
            <equals>
                <field>Enterprise.SystemSBR.Person.DOB</field>
                <source>Person.DOB</source>
            </equals>
            <equals>
                <field>Enterprise.SystemSBR.Person.Gender</field>
                <source>Person.Gender</source>
            </equals>
        </block-rule>
    </block-definition>
<config>
```

The **config** element falls within the **query-builder** element for the query being defined. Each **block-rule** element can contain multiple **equals** elements, but each **equals** element can define only one field.

3.4 Customizing the Candidate Select File

By default, two basic queries and one blocking query are predefined in the Candidate Select file. You can customize the queries used in your implementation by performing any of the following actions.

- **Defining a New Query** on page 32

- **Modifying Query Attributes** on page 33
- **Configuring Basic Queries** on page 33
- **Configuring Blocking Queries** on page 35
- **Deleting a Query** on page 39

All of these actions require that you use the Enterprise Designer XML editor. For more information about this editor, see **"Using the eView Editors" on page 21**. Make sure you are familiar with the procedures described under **"Modifying Configuration Files" on page 24** for checking the file in and out, saving changes, and validating the file.

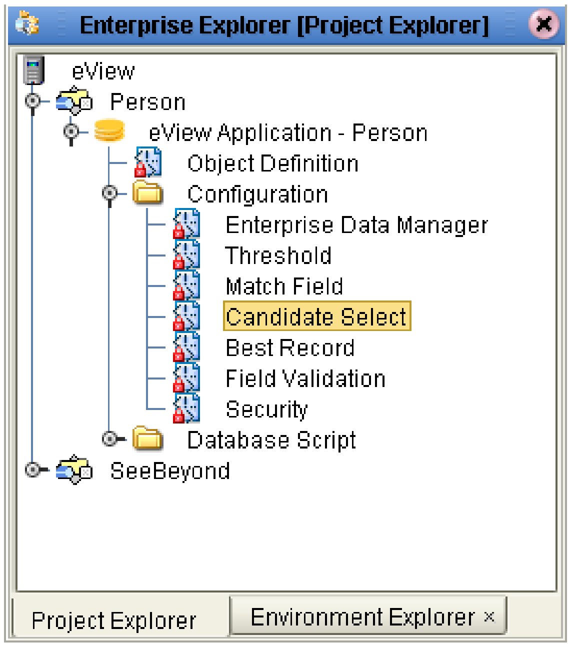
3.4.1 Defining a New Query

If the default queries provided with eView do not meet your query requirements, you can define new queries for use in your implementation. You can either use an existing query builder class or create a custom query builder through the Custom Plug-ins function (for more information, see the *eView Studio User Guide*).

To define a new query

1. In the Project Explorer pane of the Enterprise Designer, double-click **Candidate Select** under the **Configuration** folder in the Project you want to modify.

Figure 2 Candidate Select Node



2. Create a new **query-builder** element in the **QueryBuilderConfig** element.

   Make sure the new element is created outside of any existing **query-builder** elements.

3 For the new **query-builder** element, define the attributes listed in **Table 5 on page 30**. For example:

```
<query-builder name="PHONETIC-SEARCH"
class="com.stc.eindex.user.CustomQueryBuilder"
parser-class="com.stc.eindex.configurator.impl.querybuilder.
KeyValueConfiguration"
standardize="false"
phoneticize="true">
</query-builder>
```

4 Do one of the following:

- To configure a basic query, define options as described in **"Configuring Basic Queries" on page 33**.
- To configure a blocking query, define data blocks as described in **"Defining a Query Block" on page 36**.

5 Save and close the file.

3.4.2 Modifying Query Attributes

You can modify the attributes of the queries defined in the Candidate Select file if they do not meet your requirements.

To modify a query

1 In the Project Explorer pane of the Enterprise Designer, double-click **Candidate Select** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **query-builder** element that contains the query you want to modify.

3 Modify any of the attributes listed in **Table 5 on page 30**. For example:

```
<query-builder name="ALPHA-SEARCH"
class="com.stc.eindex.querybuilder.MyQueryBuilder"
parser-class="com.stc.eindex.configurator.impl.querybuilder.
KeyValueConfiguration"
standardize="true"
phoneticize="false">
</query-builder>
```

4 Save and close the file.

3.4.3 Configuring Basic Queries

Once you define a basic query, you can configure the parameters for that query. The key and value options that define parameters can also be used with any custom queries you define. Perform any of the following actions to configure default or custom basic queries:

- **Adding Parameters to a Basic Query** on page 34
- **Modifying Parameters in a Basic Query** on page 34
- **Deleting Parameters from a Basic Query** on page 35

Adding Parameters to a Basic Query

If you define a new default basic query in the Candidate Select file, you must configure the query by defining the **UseWildCard** key and value options. If you define a custom query, you can use key and value options to define any required parameters.

To add parameters to a basic query

1. In the Project Explorer pane of the Enterprise Designer, double-click **Candidate Select** under the **Configuration** folder under the node in the Project you want to modify.
2. Scroll to the **query-builder** element that contains the query you want to configure.
3. Add a new **config** element between the **query-builder** tags. For example:

```
<query-builder name="ALPHA-SEARCH"
class="com.stc.eindex.querybuilder.BasicQueryBuilder"
parser-class="com.stc.eindex.configurator.impl.querybuilder.
KeyValueConfiguration"
standardize="true"
phoneticize="true">
    <config>
    </config>
</query-builder>
```

4. In the new **config** element, create an **option** element and then define **key** and **value** attributes for the new element. For example:

```
<config>
    <option key="UseWildcard" value="true"/>
</config>
```

   See Table 7 for information about the key and value pairs available to the default basic queries.

5. Save and close the file.

Table 7 Basic Query Configuration Elements and Attributes

Element/Attribute	Description
option	One query parameter, specified by key and value attributes, as described below.
key	A value that describes the configuration option. For the default basic query, only the **UseWildCard** option is available.
value	The value of the option specified by the corresponding key attribute. For the default option, **UseWildCard**, specify **true** to allow wildcard characters for that query type; otherwise specify **false**.

Modifying Parameters in a Basic Query

Once a parameter is defined for a basic query, you can modify the name or the value of the parameter if needed. For the default basic queries, you can only modify the value.

To modify a parameter in a basic query

1. In the Project Explorer pane of the Enterprise Designer, double-click **Candidate Select** under the **Configuration** folder under the node in the Project you want to modify.
2. Scroll to the **query-builder** element that contains the query you want to configure.
3. In the **config** element of that query, change the value of the key or value attribute. For example:

```
<config>
    <option key="UseWildcard" value="false"/>
</config>
```

4. Save and close the file.

Deleting Parameters from a Basic Query

You cannot delete a parameter from the default basic queries, but you can delete parameters from the custom queries you create.

To delete a parameter from a basic query

1. In the Project Explorer pane of the Enterprise Designer, double-click **Candidate Select** under the **Configuration** folder under the node in the Project you want to modify.
2. Scroll to the **query-builder** element that contains the query you want to configure.
3. In the **config** element of that query, delete the **option** tag containing the parameter to remove. For example, to remove the SearchAlias parameter in the following sample, delete the boldface text.

```
<config>
    <option key="SearchAlias" value="true"/>
    <option key="UseWildCard" value="false"/>
</config>
```

4. Save and close the file.

3.4.4 Configuring Blocking Queries

You can customize the data blocks that are defined for the blocking queries by performing any of the following actions.

- **Defining a Query Block** on page 36
- **Deleting a Query Block** on page 37
- **Adding a Blocking Field** on page 38
- **Modifying a Blocking Field** on page 38
- **Deleting a Blocking Field** on page 39

Defining a Query Block

If you define a new blocking query in the Candidate Select file, you must configure the query by defining the data blocks to be used by the query. Each blocking query must contain at least one data block and can contain several.

To define a query block for a blocking query

1. In the Project Explorer pane of the Enterprise Designer, double-click **Candidate Select** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **query-builder** element that contains the blocking query to configure.
3. Create a new **config** element between the **query-builder** tags.
4. In the new **config** element, create a new **block-definition** element with a **number** attribute and assign a unique ID code to the **number** attribute. For example:

```
<query-builder name="BLOCKING-SEARCH"
class="com.stc.eindex.querybuilder.BlockingQueryBuilder"
parser-class="com.stc.eindex.configurator.impl.querybuilder.
KeyValueConfiguration"
standardize="true"
phoneticize="true">
    <config>
        <block-definition number="ID1">
        </block-definition>
    <config>
</query-builder>
```

5. In the new **block-definition** element, create a **block-rule** element.
6. For each field in the data block, create a new **equals** element within the new **block-rule** element, and then define a new **field** and **source** element (as shown below).

```
<config>
    <block-definition number="ID1">
     <block-rule>
        <equals>
         <field>Enterprise.SystemSBR.Person.FirstNamePh</field>
         <source>Person.FirstNamePh</source>
        </equals>
        <equals>
         <field>Enterprise.SystemSBR.Person.LastNamePh</field>
         <source>Person.LastNamePh</source>
        </equals>
     </block-rule>
    </block-definition>
<config>
```

Note: *The available attributes and elements within a blocking query **config** element are listed in **Table 8 on page 37**.*

7. Repeat steps 4 through 6 for each data block you need to define for the query.

   All data blocks for one query must be defined within one **config** element.
8. Save and close the file.

Table 8 Blocking Query Configuration Elements

Elements	Description
block-definition	A list of defined query criteria blocks, identified by unique ID numbers.
number	An attribute of the **block-definition** element that specifies the unique ID number of each query block. Each block defined for the block query must be identified by a unique ID.
block-rule	A list of fields to be included in each query block.
equals	A field definition for one field in a **block-rule** element.
field	The fully qualified field name of the field to be included in the query block (for example, **Enterprise.Person.Address.AddressLine1**).
source	The qualified field name of the source field in the object from which the criteria is obtained (for example, **Person.Address.AddressLine1**).

Deleting a Query Block

If a data block is defined in error for a blocking query, or is obsolete, you can remove that block from the query configuration.

To delete a query block from a blocking query

1. In the Project Explorer pane of the Enterprise Designer, double-click **Candidate Select** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **query-builder** element that contains the blocking query to configure.
3. In that **query-builder** element, delete all text between and including the **block-definition** element defining the data block to remove.

   For example, in the sample below, to delete the SSN block you would delete the boldface text.

```
<config>
    <block-definition number="ID000007">
        <block-rule>
            <equals>
                <field>Enterprise.SystemSBR.Person.SSN</field>
                <source>Person.SSN</source>
            </equals>
        </block-rule>
    </block-definition>
    <block-definition number="ID000008">
        <block-rule>
            <equals>
             <field>Enterprise.SystemSBR.Person.DOB</field>
                <source>Person.DOB</source>
            </equals>
        </block-rule>
    </block-definition>
```

Note: *Each blocking query must contain one **config** element, and the **config** element must contain at least one defined data block.*

4 Save and close the file.

Adding a Blocking Field

Blocking queries contain predefined blocks of criteria, which are used during the match process. You can add new fields to any existing data blocks.

To add a blocking field

1 In the Project Explorer pane of the Enterprise Designer, double-click **Candidate Select** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **query-builder** element to which you want to add a blocking field.

3 Scroll to the **block-definition** element you want to modify, and then scroll to the **block-rule** element to modify.

4 Create a new **equals** element within the appropriate **block-rule** element, but outside any existing **equals** elements. For example:

```
<config>
    <block-definition number="ID000007">
        <block-rule>
            <equals>
                <field>Enterprise.SystemSBR.Person.SSN</field>
                <source>Person.SSN</source>
            </equals>
            <equals>
            </equals>
        </block-rule>
    </block-definition>
<config>
```

5 In the new **equals** element, create and define a new **field** element and a corresponding **source** element, as shown below. These elements are described in **Table 8 on page 37**.

```
<equals>
    <field>Enterprise.SystemSBR.Person.DOB</field>
    <source>Person.DOB</source>
</equals>
```

6 Save and close the file.

Modifying a Blocking Field

Blocking queries contain predefined blocks of criteria, which are used during the match process. These data blocks are based on information you specified in the eView Wizard. You can modify any of the fields defined for a blocking query.

To modify a blocking field

1 In the Project Explorer pane of the Enterprise Designer, double-click **Candidate Select** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **query-builder** element that contains the query you want to modify.

3. Scroll to the **block-definition** element you want to modify, and then scroll to the appropriate **equals** element.
4. Modify the value of the **field** or **source** element, as shown below. These elements are described in **Table 8 on page 37**.

```
<equals>
    <field>Enterprise.SystemSBR.Person.MStatus</field>
    <source>Person.Mstatus</source>
</equals>
```

5. Save and close the file.

Deleting a Blocking Field

You can delete fields defined in a data block if they are not needed for the blocking query. Each data block must contain at least one field definition.

To delete a blocking field

1. In the Project Explorer pane of the Enterprise Designer, double-click **Candidate Select** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **query-builder** element that contains the field you want to delete, and then scroll to the **block-definition** element to modify.
3. Delete all text between and including the **equals** tags that define the blocking field to remove.

   For example, using the sample below, to delete the first name field from the block, delete the boldface text.

```
<block-rule>
    <equals>
        <field>Enterprise.SystemSBR.Person.LastName/field>
        <source>Person.LastNamePhonetic</source>
    </equals>
    <equals>
        <field>Enterprise.SystemSBR.Person.FirstName</field>
        <source>Person.FirstNamePhonetic</source>
    </equals>
</block-rule>
```

Note: *Each **block-rule** element must contain at least one **equals** element. To delete all fields from a block rule, delete the entire block as described in **"Deleting a Query Block" on page 37**.*

4. Save and close the file.

3.4.5 Deleting a Query

If a query is defined that will not be used in the master index, you can delete that query. However, you can leave the query in the Candidate Select file as it will not affect processing if you do not delete the query.

To delete query

1. In the Project Explorer pane of the Enterprise Designer, double-click **Candidate Select** under the **Configuration** folder under the node in the Project you want to modify.
2. Scroll to the **query-builder** element that contains the query you want to remove.
3. Delete all text between and including the **query-builder** element defining the query. This includes any defined query blocks. For example, to delete a phonetic search, remove all text below.

```
<query-builder name="PHONETIC-SEARCH"
    class= "com.stc.eindex.querybuilder.BasicQueryBuilder"
    parser-class= "com.stc.eindex.configurator.impl.querybuilder.
    KeyValueConfiguration"
    standardize="true" phoneticize="true">
        <config>
            <option key="UseWildcard" value="false"/>
        </config>
</query-builder>
```

4. Save and close the file.

Chapter 4

Object Definition

After you define the eView instance and create the configuration files, you can modify the object structure that you defined. This chapter describes the Object Definition file, and provides instructions for modifying the objects, fields, and relationships of the index.

4.1 About the Object Definition

The properties for the objects you will store in the eView database are defined in the Object Definition XML file. This file defines the parent and child objects to be indexed, the fields contained in each object, along with key properties for each field, such as the field size, unique record identifiers, and whether certain fields are required or can be updated.

The Object Definition is used as a basis for most of the master index. The information you specify for this file defines the method Object Type Definition (OTD) and the database structure for the primary tables that store object information in the master index.

4.2 Object Definition Components

The Object Definition contains three primary components.

- Objects
- Fields
- Relationship definitions

Together, these three components define the structure of the data in the master index, the database structure, and the eGate method OTD. Most configuration files in the eView system rely on the objects and fields defined in the Object Definition. For example, the fields you specify for the match string, queries, standardization, and the survivor calculator must all be defined in the Object Definition.

4.2.1 Objects

In eView, information is stored in objects. Each object in the data structure represents a different type of information. For example, if you are indexing businesses, you might have one object type to store general information about the business (such as the business name and type), one to store address information, and one to store telephone information. The object structure can have several objects, but only one primary object (called the parent object). This object is the parent to all other objects defined in the Object Definition. The object structure can have multiple child objects.

Generally, a record in the master index has information in one parent object and multiple child objects. A record can also have multiple instances of each child object. For example, in the business index example above, a record for a single business would have one business name and one business type, both stored in the parent object. However, the same record might have several different addresses, each of which is stored in a separate Address object.

4.2.2 Fields

Each object in the object structure contains fields that store each data element in the object. You can specify properties for each field in the object structure, such as a length, name, data type, formatting rules, and so on. The fields you define in the object structure help determine the structure of the method OTD and the database tables. You can also specify certain properties for each field that determine how the database columns are defined, including the length, name, and required data type.

4.2.3 Relationships

In the Object Definition, you must specify the parent and child objects. The object structure must contain one parent object. All remaining objects defined in the structure must be specified as child objects to that parent object.

4.3 The Object Definition File

The object structure is defined in the Object Definition file in XML format. The information entered into the default configuration file is based on the objects and fields you defined in the eView Wizard. Depending on how completely you defined the object structure in the eView Wizard, this file should not require customization.

4.3.1 Modifying the Object Definition

When you use the eView Wizard to define the object structure, all the configuration files for the master index are automatically generated, based on the information you provide. You can modify this file at any time prior to deploying the associated Project, but you must update the remaining configuration files, regenerate the Project, and reactivate the deployment after doing so. For example, if you delete a field from Object

Definition that also appears on the EDM, appears in the SBR, and is defined for standardization and matching, you must remove the field from the Enterprise Data Manager file, Best Record file, and Match Field file. Any changes made to the file without regenerating the Project will not take effect. You can either apply the changes immediately to the Logical Host, or restart the host to pick up the changes.

Before making any changes to this file, make sure you are familiar with the procedures described under **"Modifying Configuration Files" on page 24**. The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes.

4.3.2 Object Definition File Structure

The Object Definition file is written in XML, and is automatically generated by the eView Wizard. This file can be modified using the Enterprise Designer XML editor, which is a standard NetBeans editor. This section describes the structure of the XML file and general requirements and constraints. There are five primary components of the Object Definition file. The first component, XML File Information, is standard across all implementations.

- XML File Information
- Deployment Environment Information
- Objects
- Fields
- Relationships

XML File Information

The first lines in the Object Definition XML file represent information about the schema instance and namespace. You should not need to modify these components.

Deployment Environment Information

Following the XML-specific information are three elements that describe the deployment environment. They include the name of the eView instance, the database platform, and the date format to use for the application. Below is a sample of this information.

```
<name>Company</name>
<database>Oracle</database>
<dateformat>MM/dd/yyyy</dateformat>
```

Note: *Three formats are allowed for the date:* ***MM/dd/yyyy***, ***yyyy/MM/dd***, *and* ***dd/MM/yyyy***. *The name of the application must match the name of the parent object.*

Objects

The Object Definition file defines the types of objects stored in the database, along with each data field contained in each object type. For example, if you are storing information about people, you may want to store demographic data, addresses, and telephone numbers. Your object types might be Person, Address, and Telephone. One of the objects must be the parent object, with the remaining objects being the child objects (for more information, see **"Relationships" on page 45**). The name of the parent object must match the name of the application.

Objects and their associated fields are defined within **nodes** elements, and the object names are defined within **tag** elements. Below is a sample defining three object types.

```
<nodes>
    <tag>Person</tag>
</nodes>
<nodes>
    <tag>Address</tag>
</nodes>
<nodes>
    <tag>Telephone</tag>
</nodes>
```

Fields

For each object type you define, you must also define and configure the fields that belong to the object. Fields are specified within **fields** elements that are inserted immediately following the object's **tag** element. Fields contained in a specific object must be defined within the **nodes** tags for that object. There are several properties you can configure for each field, including the field name, data type, field length, whether the field can be updated, whether the field is required, maximum and minimum field values, and so on.

For a complete list of the field properties that you can configure for each field, see **Table 9 on page 52**.

Important Field Name Restrictions:

- *A field is automatically created for each object named **<object>Id**, where **<object>** is the name of the object or sub-object. You cannot create additional fields by those names. For example, you cannot create a field named "AddressId" if there is an Address object in the object structure.*
- *Field names cannot be longer than 30 characters.*

The following sample illustrates a definition for a field named "Name" that is contained in a Company object.

```
<nodes>
    <tag>Company</tag>
    <fields>
        <field-name>Name</field-name>
        <field-type>string</field-type>
        <size>100</size>
        <updateable>true</updateable>
        <required>true</required>
        <code-module/>
        <pattern/>
        <key-type>false</key-type>
    </fields>
<nodes>
<nodes>
    <tag>Address</tag>
</nodes>
```

Relationships

In the Object Definition file, relationships define the hierarchy of the object types you defined in the **nodes** elements. By specifying relationships, you define parent and child objects. Only one object can be the parent object, and the remaining objects must be defined as child objects of the parent object. Relationships allow the parent object to contain more than one instance of each child object. For example, if you define the parent object as a Person, with Address and Telephone child objects, then each Person object can contain multiple addresses and telephone numbers. Define child objects for any type of information that may occur in multiple instances in the parent object.

Relationships are defined after the **nodes** elements and within a **relationships** element. Within the **relationships** element, the parent node is defined in a **name** element, and the child nodes are defined in **children** elements. The names you specify for the objects in this section must match the names you specified for the objects defined in the **nodes** elements earlier in the file. In the following sample, the parent object is “Company”, which contains the child objects “Phone” and “Address”.

```
<relationships>
    <name>Company</name>
    <children>Phone</children>
    <children>Address</children>
</relationships>
```

4.4 Customizing the Object Definition

Once you complete the eView Wizard and define the basic structure of the object, you can further customize the Object Definition file as needed to configure the object to fit your data requirements. Only changes made before generating the Project take effect for the new application. To modify the object from its default configuration, you can perform the following actions.

- **Creating a New Object** on page 46
- **Modifying an Object Name** on page 47
- **Deleting an Object** on page 47

- **Adding a Field to an Object** on page 48
- **Deleting a Field from an Object** on page 48
- **Modifying Field Properties** on page 49
- **Defining Relationships Between Objects** on page 53

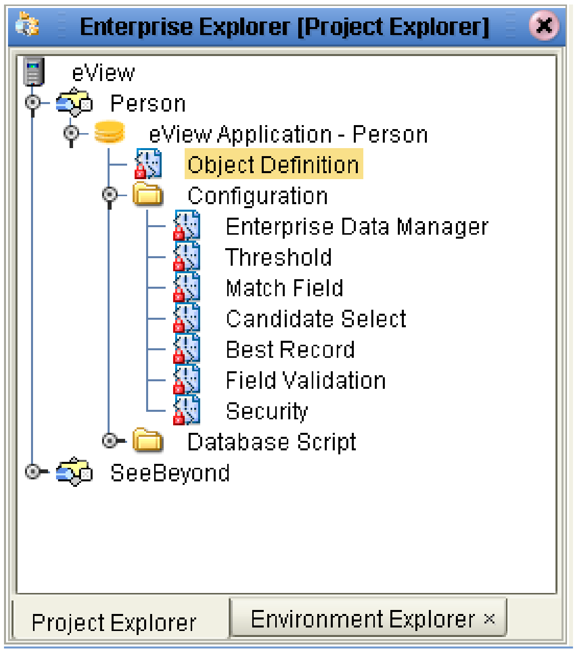
All of these actions require that you use the Enterprise Designer XML editor. For more information about this editor, see **"Using the eView Editors" on page 21**. Make sure you are familiar with the procedures described under **"Modifying Configuration Files" on page 24** for checking the file in and out, saving changes, and validating the file.

Important: *If you modify the fields and objects in this file, you must also modify the remaining configuration files that reference those fields or objects.*

4.4.1 Creating a New Object

You can add new objects to the object structure as needed. Note that each Object Definition file can contain only one parent object.

Figure 3 Object Definition Node



To create a new object

1. In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.
2. Scroll to the location where you want to create the new object (after the **database** element but before **relationships**).
3. Create a **nodes** element, and then create and name a **tag** element within the **nodes** element (the value of the **tag** element is the name of the object).

Make sure the new **nodes** element does not fall within any existing **nodes** elements. For example:

```
<nodes>
    <tag>Company</tag>
        ...
</nodes>
<nodes>
    <tag>Address</tag>
</nodes>
```

4 Define the fields for the new object, as described in **"Adding a Field to an Object" on page 48**.

5 Define the relationship of the new object to the existing objects, as described in **"Defining Relationships Between Objects" on page 53**.

6 Save and close the file.

4.4.2 Modifying an Object Name

Once you define an object in the Object Definition file, you can modify the name of the object. Remember to make the corresponding changes to the remaining configuration files. The name of the parent object must match the name of the application.

To modify an object name

1 In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.

2 Scroll to the **tag** element defining the object you want to modify.

3 Change the value of the **tag** element. For example:

```
<tag>Business</tag>
```

4 Save and close the file.

4.4.3 Deleting an Object

If you define an object in error, you can remove the object from the Object Definition file. You must also remove the relationship definition for the object. Remember to make the corresponding changes to the remaining configuration files.

To delete an object

1 In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.

2 Scroll to the **nodes** element containing the object you want to delete.

3 Delete all text between and including the **nodes** tags that contain the object tag. For example, to delete the Address object below, delete the boldface text.

```
<nodes>
    <tag>Business</tag>
</nodes>
<nodes>
    <tag>Address</tag>
</nodes>
```

4 Remove the object name from the relationship list, as described in **"Defining Relationships Between Objects" on page 53**.

5 Save and close the file.

4.4.4 Adding a Field to an Object

Once you define an object in the Object Definition file, you can add new fields to the object and configure certain properties for those fields.

To add a field to an object

1 In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.

2 Scroll to the **tag** element defining the object to which you want to add fields.

3 Under the **tag** element, create a new **fields** element. For example:

```
<nodes>
    <tag>Address</tag>
    <fields>
    </fields>
<nodes>
```

4 Specify the field properties described in **Table 9 on page 52** within the new **fields** tags. For example:

```
<fields>
    <field-name>AddressType</field-name>
    <field-type>string</field-type>
    <size>8</size>
    <updateable>true</updateable>
    <required>true</required>
    <code-module>ADDRTYPE</code-module>
    <pattern/>
    <key-type>true</key-type>
</fields>
```

5 Save and close the file.

4.4.5 Deleting a Field from an Object

If a field is defined for an object but does not belong to that object, you can delete the field from the object structure. Remember to make the corresponding changes to the remaining configuration files.

To delete a field from an object

1 In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.

2 Scroll to the **tag** element defining the object from which you want to delete a field.

3 Scroll to the **fields** element containing the field to delete, and then delete all text between and including the **fields** tags defining that field.

For example, to delete the AddressLine1 fields below, delete all text in the sample.

```
<fields>
    <field-name>AddressLine1</field-name>
    <field-type>string</field-type>
    <size>5</size>
    <updateable>true</updateable>
    <required>false</required>
    <key-type>false</key-type>
    <visible>true</visible>
</fields>
```

4 Save and close the file.

4.4.6 Modifying Field Properties

Once a field is defined in the Object Definition file, you can modify the properties for that field. Refer to **Table 9 on page 52** for more information about the elements that define the field properties.

To change a field name

1 In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.

2 Scroll to the **tag** element defining the object to modify, and then to the **fields** element containing the field to modify.

3 Modify the value of the **field-name** element as shown below.

```
<field-name>StreetAddress</field-name>
```

4 Save and close the file.

Note: *Remember to make the corresponding changes to the remaining configuration files.*

To change a field type

1 In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.

2 Scroll to the **tag** element defining the object to modify, and then to the **fields** element containing the field to modify.

3 Modify the value of the **field-type** element as shown below.

```
<field-type>string</field-type>
```

4 Save and close the file.

To change the length of a field

1 In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.

2 Scroll to the **tag** element defining the object to modify, and then to the **fields** element containing the field to modify.

3 Modify the value of the **size** element as shown below.

```
<size>100</size>
```

4 Save and close the file.

To specify whether a field can be modified in the database

1. In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.
2. Scroll to the **tag** element defining the object to modify, and then to the **fields** element containing the field to modify.
3. Modify the value of the **updateable** element as shown below.

   ```
   <updateable>true</updateable>
   ```
4. Save and close the file.

To specify whether a field is required

1. In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.
2. Scroll to the **tag** element defining the object to modify, and then to the **fields** element containing the field to modify.
3. Modify the value of the **required** element as shown below.

   ```
   <required>false</required>
   ```
4. Save and close the file.

To specify a drop-down menu for a field

1. In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.
2. Scroll to the **tag** element defining the object to modify, and then to the **fields** element containing the field to modify.
3. Modify the value of the **code-module** element as shown below.

   ```
   <code-module>ADDRTYPE</code-module>
   ```

Note: *This value must correspond to a value in the **code** column of the sbyn_common_header database table unless the drop-down list is populated from the sbyn_user_code table (as would be the case for auxiliary IDs). In this case, it must match a value in the **code-list** column of sbyn_user_code.*

4. Save and close the file.

To specify a minimum value for a field

1. In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.
2. Scroll to the **tag** element defining the object to modify, and then to the **fields** element containing the field to modify.
3. Modify the value of the **minimum-value** element as shown below.

   This sample specifies that the date of birth can be no earlier than 01/01/1850.

   ```
   <field-name>DateOfBirth</field-name>
   <minimum-value>1850-01-01</minimum-value>
   ```
4. Save and close the file.

To specify a maximum value for a field

1. In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.
2. Scroll to the **tag** element defining the object to modify, and then to the **fields** element containing the field to modify.
3. Modify the value of the **maximum-value** element as shown below.

   This sample specifies that the PaymentDate can be no later than 12/31/2003.

   ```
   <field-name>PaymentDate</field-name>
   <maximum-value>2003-12-31</maximum-value>
   ```
4. Save and close the file.

To specify a format for a field

1. In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.
2. Scroll to the **tag** element defining the object to modify, and then to the **fields** element containing the field to modify.
3. Modify the value of the **pattern** element as shown below.

   This sample specifies that the TelephoneNumber field can contain the numbers 0 through 9, and must contain 10 characters.

   ```
   <field-name>TelephoneNumber</field-name>
   <pattern>[0-9]{10}</pattern>
   ```

Note: *For information about the types of patterns you can use, see "Patterns" in the class list for **java.util.regex** in the Javadocs provided with J2SDK.*

4. Save and close the file.

To specify whether a field must be unique to the parent object

1. In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.
2. Scroll to the **tag** element defining the object to modify, and then to the **fields** element containing the field to modify.
3. Modify the value of the **key-type** element as shown below.

   Specify **true** to make the field unique; specify **false** if the field does not need to be unique.

   ```
   <key-type>true</key-type>
   ```
4. Save and close the file.

To specify that a combination of fields must be unique to the parent object

1. In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.
2. Scroll to the **tag** element defining the object to modify, and then to the **fields** element containing the first field in the unique field combination.

3 Enter **true** for the **key-type** element. For example:

```
<key-type>true</key-type>
```

4 Repeat steps 2 and 3 for each field you want to include in the unique key combination.

5 Save and close the file.

Table 9 Object Definition Field Properties

Element	Description
field-name	The name of the field. This element must conform to Oracle naming standards for database columns and cannot be longer than 30 characters.
field-type	The data type for each field, such as string, object, date, and so on. Possible values for this field are **string**, **int**, **long**, **char**, **float**, **date**, or **boolean**.
size	The number of characters allowed in each field. This defines the length of the corresponding database column.
updateable	An indicator of whether the field can be updated via the EDM or back-end messages. Specify **true** if the field can be updated, or specify **false** if it cannot.
required	An indicator of whether the field is required in order to save an enterprise object in the database. Specify **true** if the field is required, or specify **false** if it is not.
code-module	The identification code for the menu list that will appear for this field in the EDM. This must match a value in the **code** column of the sbyn_comment_header database table. This element is optional.
maximum-value	The maximum value allowed in the field. To specify a value for a date field, use the format "YYYY-MM-DD". This element is optional.
minimum-value	The minimum value allowed in the field. To specify a value for a date field, use the format "YYYY-MM-DD". This element is optional.
pattern	The required pattern for the field. For more information about possible values and using Java patterns, see "Patterns" in the class list for **java.util.regex** in the Javadocs provided with J2SDK. This element is optional.
key-type	An indicator of whether the field is used to identify unique objects. Specify **true** if the element is a unique record identifier; specify **false** if it is not. This element is optional. ***Note:*** *Each child object must contain at least one field that is a unique record identifier. If two or more fields are unique identifiers, the combined value of these fields must be unique in a given record.*

4.4.7 Defining Relationships Between Objects

Once all objects are customized, you must define relationships between those objects.

To define object relationships

1. In the Project Explorer pane of the Enterprise Designer, double-click the **Object Definition** node in the Project you want to modify.
2. Scroll to the **relationships** element near the end of the file.
3. To specify a new parent object, modify the value of the **name** element. For example:

   ```
   <name>Individual</name>
   ```

4. To change the name of a child object, modify the value of the appropriate **children** element. For example:

   ```
   <children>Telephone</children>
   ```

5. To add a child object, create and name a new **children** element. For example:

   ```
   <children>Telephone</children>
   <children>AddressData</children>
   ```

6. To delete a child object, delete all text between and including the appropriate **children** element.

   For example, to delete the AddressData child object above, delete the following text: <children>AddressData</children>.

7. Save and close the file.

Note: *You can only specify one **name** element. The values you specify for the **name** and **children** elements must match an object name specified in the **nodes** elements earlier in the file.*

Chapter 5

Threshold Configuration

The Threshold file defines certain system parameters, such as matching thresholds and EUID properties, for the eView Manager Service. It also specifies the blocking query to use for match processing. This chapter describes the eView Manager Service and the Threshold file, and provides instructions for customizing the eView Manager Service for your processing requirements.

5.1 About Threshold Configuration

The Threshold file specifies information for the main interface of the indexing system, the eView Manager Service. This interface coordinates all components of the master index, including the database, eView Project, Enterprise Data Manager, runtime environment, and match engine. The main interface is a stateless session bean, though some methods return objects that have handles to stateful beans.

5.2 eView Manager Service Components

The Threshold file defines certain properties of the match process, such as duplicate and match thresholds, the query to use for matching, logic for automatic merges, and properties of the EUIDs assigned by eView (such as their length and whether a checksum value is used). It also defines the update mode (optimistic or pessimistic) and merged record updates. The following eView Manager Service components are configured by the Threshold file:

- Master Controller Configuration
- Decision Maker
- EUID Generator

5.2.1 Master Controller Configuration

The **MasterControllerConfig** element of the Threshold file controls three components of the matching and update process.

- Update Mode

- Merged Record Updates
- Blocking Query

Update Mode

The update mode specifies whether a record's potential duplicate list is re-evaluated when key fields are updated in the record. Performing the re-evaluation helps keep the potential duplicate list current, but requires more system resources. There are two update modes.

- **Pessimistic**
  In this mode, a record's potential duplicates are re-evaluated whenever updates are made to the record's key fields. Key fields are fields involved in blocking and matching.
- **Optimistic**
  In this mode, potential duplicates are not re-evaluated when key fields are updated in a record. After an update, the potential duplicate list for a record remains the same as before the update occurred.

Merged Record Updates

The merge update status determines whether changes can be made to records that have a status of "merged". These are the EUID records that are not retained after a merge. For example, when an incoming record is an assumed match with an SBR that has a status of "merged", the master index checks the value of the **merged-record-update** element. If the element is set to "Enabled", the merged SBR is updated with the new information. If the element is set to "Disabled", an exception is thrown and the update is not performed. Typically, it is recommended that merged records not be updated.

Blocking Query

The blocking query, specified by the **query-builder** element, identifies one of the queries defined in the Candidate Select file as the query to use for match processing. This query is used by the master index when searching for a candidate pool of possible matches to an incoming record. If the query takes any parameters, they are defined using the **option** element.

5.2.2 Decision Maker

The **DecisionMakerConfig** element of the Threshold file allows you to specify how the eView Manager Service evaluates query results. For the default Decision Maker, you can configure these parameters:

- OneExactMatch
- SameSystemMatch
- DuplicateThreshold
- MatchThreshold

When the master index processes an incoming record, it compares the new record against existing records in the database and assigns a matching weight between possible matches with the incoming record. The master index uses the values that you specify in this section to determine how to handle records that fall within certain matching weight ranges. Records with a matching weight above the duplicate threshold are treated as potential duplicates; records with a matching weight above the match threshold are treated as potential duplicates or assumed matches, depending on the value of the **OneExactMatch** parameter and the number of records with a matching weight above the match threshold.

OneExactMatch

This parameter specifies logic for assumed matches. If **OneExactMatch** is set to **true** and there is more than one record above the match threshold, then none of the elements are considered an assumed match and all are flagged as potential duplicates.

SameSystemMatch

This parameter indicates whether the master index will match two records that originated from the same system whose matching weight falls above the match threshold. If **SameSystemMatch** is set to **true**, no assumed matches are made between records associated with the same system.

DuplicateThreshold

The duplicate threshold specifies the matching probability weight at or above which two records are considered to be potentially represent the same object. Records with matching weights between the duplicate and match thresholds are always flagged as potential duplicates. A thorough data analysis combined with testing will help determine the best value for the duplicate and match thresholds.

MatchThreshold

The match threshold specifies the matching probability weight at or above which two records are assumed to be a match and are automatically merged in the master index database.

5.2.3 EUID Generator

The EUID generator controls how EUIDs are created for each unique record in the master index database. For the default EUID generator, you can define three parameters.

- IdLength
- ChecksumLength
- ChunkSize

IdLength

This parameter defines the length of the EUIDs created by the master index. By default, the length of the EUID columns in the master index database is 20. If you choose an ID length larger than 20, make sure to manually modify the length of the EUID columns in the database creation scripts. These scripts are described in the *eView Studio User's Guide*.

ChecksumLength

The **ChecksumLength** parameter allows you to specify the length of a checksum value. Checksum values help validate EUIDs to ensure accurate identification of records as they are transmitted throughout the system. The checksum process attaches a number, generated through an algorithm, to the end of a new EUID. When a host system receives this number, it strips off the checksum digits to obtain the EUID, and then recalculates the checksum using the same algorithm process. If the checksums agree, the host system knows the EUID number is correct. Specify "0" (zero) if you do not want to use the checksum function.

Checksum Values and EUID Lengths

Using a checksum value affects the **IdLength** parameter. If you specify a checksum length greater than 0, the EUID generator creates sequential EUIDs based on the sbyn_seq_table table, and then appends the checksum value to the end of the EUID to determine the final EUID number. For example, if you set **IdLength** to 8 and **CheckSum** to 2, then the EUIDs assigned by the master index will be 10 characters long. If the next sequence number is 10908000, the EUID assigned to the next record is 10908000 plus the checksum (it might be 1090800034, for example). The next EUID would be 10908001 plus the checksum (1090800125, for example). The first eight digits are sequential, but the last two digits are seemingly arbitrary.

If you use a checksum value, make sure to take into consideration the total length of the EUIDs (**IdLength** plus **ChecksumLength**) when determining the length of the EUID columns in the database.

ChunkSize

For efficiency, the default EUID generator does not need to query the sbyn_seq_table table in the database each time a new EUID is created. Instead, you can specify a number of EUIDs to be allocated in chunks to the EUID generator. For example, if you specify a chunk size of 1000, EUIDs are allocated to the generator 1000 ID numbers at a time. The generator can process up to 1000 new records and assign all 1000 numbers without needing to query sbyn_seq_table. When all 1000 EUIDs are used, another 1000 are allocated. If the server storing the eView files is reset before all 1000 numbers are used, the unused numbers are discarded and never used, meaning that EUIDs may not always be assigned sequentially.

Specifying a chunk size affects the numbering of the EUID column in the sbyn_seq_table. If you specify a chunk size of "1", then each time a new EUID is assigned, the value of the EUID column increases by one. If you specify a larger chunk size, then the value of the EUID column increases by the value of the chunk size each time the allocated EUIDs are used. For example, if you specify a chunk size of 1000, the beginning EUID sequence number is 1000, even though EUIDs are assigned beginning with “0001”, then “0002”, and so on. When the first 1000 EUIDs are assigned, another 1000 EUID numbers are allocated to the generator and the EUID column changes from 1000 to 2000.

5.3 The Threshold File

The properties of the eView Manager Service are defined in the Threshold file in XML format. The information entered into the default configuration file is standard across all implementations, so the file will require some customization.

5.3.1 Modifying the Threshold File

You can modify the Threshold file at any time, but you must regenerate the Project and reactivate the deployment after making any changes to the file. You can either apply the changes immediately to the Logical Host, or restart the host to pick up the changes. Use caution when updating this file after moving into production, since changing certain properties, such as the blocking query, can cause unexpected matching and weighting results.

Before making any changes to this file, make sure you are familiar with the procedures described under **“Modifying Configuration Files” on page 24**. The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes.

5.3.2 Threshold File Structure

The Threshold file is written in XML, and is automatically generated by the eView Wizard. This file can be modified using the Enterprise Designer XML editor, which is a standard NetBeans editor. This section describes the structure of the XML file and general requirements and constraints. There are four primary components of the configuration file. The first component, XML File Information, is standard across all implementations.

- XML File Information
- Master Controller Configuration
- Decision Maker
- EUID Generator

XML File Information

The first lines in the Threshold file represent information about the schema instance and namespace. You should not need to modify these components.

Master Controller Configuration

The Master Controller configuration section of the Threshold file is specified by the **MasterControllerConfig** element, and begins by defining a module name and parsing class for the module. These two elements should not require any changes. The **update-mode** element specifies whether potential duplicates are re-evaluated on updates and the **merged-record-update** element specifies whether merged records can be updated. These elements are shown below.

```
<update-mode>Pessimistic</update-mode>
<merged-record-update>Disabled</merged-record-update>
```

The **execute-match** element specifies the blocking query to use for match processing. The blocking query is defined by the **name** attribute of the **query-builder** element in **execute-match**. Optionally, you can specify key and value options for the blocking query, but currently these options are not used by any predefined blocking queries. The name specified for the **query-builder** element must match a query defined in the Candidate Select file. An example of the **execute-match** element is shown below.

```
<execute-match>
    <query-builder name="BLOCKER-SEARCH">
        <option key="key" value="value"/>
    </query-builder>
</execute-match>
```

Note: *The BLOCKER-SEARCH query does not use **option** elements by default, however it is shown here for example.*

Decision Maker

The Decision Maker section of the Threshold file is specified by the **DecisionMakerConfig** element, and begins by defining a module name and parsing class for the module. These two elements should not require any changes. The Java class that contains the methods used by the Decision Maker is specified by the element **decision-maker-class**. The default class is **com.stc.eindex.decision.impl.DefaultDecisionMaker**, which accepts the parameters defined in **"Decision Maker" on page 55**. You can also implement a custom Decision Maker class.

The parameters are all defined within one **parameters** element, and each parameter is configured within its own **parameter** element, as shown below. For a complete list of the elements that define a parameter, see **Table 10 on page 68**.

```
<decision-maker-class>
com.stc.eindex.decision.impl.DefaultDecisionMaker
</decision-maker-class>
        <parameters>
            <parameter>
                <parameter-name>OneExactMatch</parameter-name>
                <parameter-type>java.lang.Boolean</parameter-type>
                <parameter-value>false</parameter-value>
            </parameter>
            <parameter>
                <parameter-name>SameSystemMatch</parameter-name>
                <parameter-type>java.lang.Boolean</parameter-type>
                <parameter-value>true</parameter-value>
            </parameter>
```

```
                <parameter>
                    <parameter-name>DuplicateThreshold</parameter-name>
                    <parameter-type>java.lang.Float</parameter-type>
                    <parameter-value>7.25</parameter-value>
                </parameter>
                <parameter>
                    <parameter-name>MatchThreshold</parameter-name>
                    <parameter-type>java.lang.Float</parameter-type>
                    <parameter-value>29.0</parameter-value>
                </parameter>
            </parameters>
        </DecisionMakerConfig>
```

EUID Generator

The EUID Generator section of the Threshold file is specified by the **EuidGeneratorConfig** element, and begins by defining a module name and parsing class for the module. These two elements should not require any changes.

The Java class that contains the methods used by the EUID Generator is defined in the **euid-generator-class** element. The default generator class is named **com.stc.eindex.idgen.impl.DefaultEuidGenerator**, and assigns sequential EUIDs based on the three parameters described in **"EUID Generator" on page 56**. The parameters are all defined within one **parameters** element, and each parameter is configured within its own **parameter** element, as shown below. For a complete list of parameters, see **Table 10 on page 68**.

```
<euid-generator-class>com.stc.eindex.idgen.impl.DefaultEuidGenerator
</euid-generator-class>
    <parameters>
        <parameter>
            <parameter-name>IdLength</parameter-name>
            <parameter-type>java.lang.Integer</parameter-type>
            <parameter-value>10</parameter-value>
        </parameter>
        <parameter>
            <parameter-name>ChecksumLength</parameter-name>
            <parameter-type>java.lang.Integer</parameter-type>
            <parameter-value>2</parameter-value>
        </parameter>
        <parameter>
            <parameter-name>ChunkSize</parameter-name>
            <parameter-type>java.lang.Integer</parameter-type>
            <parameter-value>1000</parameter-value>
        </parameter>
    </parameters>
</EuidGeneratorConfig>
```

5.4 Customizing the Threshold File

You can customize the Threshold file by performing any of the following actions.

- **Configuring the Master Controller** on page 61
- **Configuring the Decision Maker** on page 64
- **Configuring the EUID Generator** on page 69

All of these actions require that you use the Enterprise Designer XML editor. For more information about this editor, see **"Using the eView Editors" on page 21**. Make sure you are familiar with the procedures described under **"Modifying Configuration Files" on page 24** for checking the file in and out, saving changes, and validating the file.

5.4.1 Configuring the Master Controller

The Master Controller section allows you to specify the blocking query to use when the master index queries the database to find a candidate selection pool for an incoming record and to set options for that search. You can also specify whether merged records can be updated, and whether a record's potential duplicates are re-evaluated after it is updated. Customize the Master Controller section by performing any of the following actions:

- **Defining the Update Mode** on page 61
- **Configuring Merged Record Updates** on page 62
- **Specifying the Blocking Query for Matching** on page 63
- **Setting Blocking Query Options** on page 63

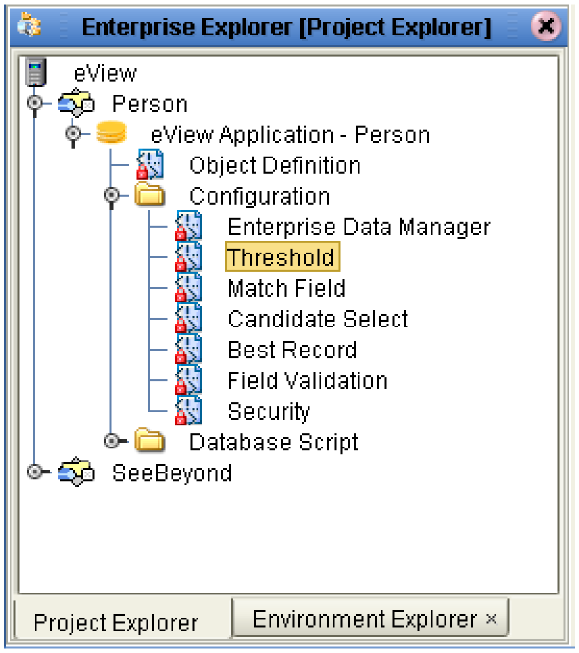
Defining the Update Mode

The **update-mode** element specifies whether potential duplicates are re-evaluated for a record each time the record is updated. If you specify that potential duplicates are re-evaluated, the re-evaluation only occurs when updates are made to fields involved in blocking and matching.

To specify potential duplicates be re-evaluated at each update

1 In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.

Figure 4 Threshold Node



2 In the **MasterControllerConfig** element, change the value of the **update-mode** element to “Pessimistic”. For example:

```
<update-mode>Pessimistic</update-mode>
```

3 Save and close the file.

To specify potential duplicates not be re-evaluated at each update

1 In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.

2 In the **MasterControllerConfig** element, change the value of the **update-mode** element to “Optimistic”. For example:

```
<update-mode>Optimistic</update-mode>
```

3 Save and close the file.

Configuring Merged Record Updates

The **merged-record-update** element allows you to define whether updates can be made to records with a status of “merged”.

To allow merged record updates

1 In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.

2 In the **MasterControllerConfig** element, change the value of the **merged-record-update** element to “Enabled”. For example:

```
<merged-record-update>Enabled</merged-record-update>
```

3 Save and close the file.

To prevent merged record updates

1 In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.

2 In the **MasterControllerConfig** element, change the value of the **merged-record-update** element to "Disabled". For example:

```
<merged-record-update>Disabled</merged-record-update>
```

3 Save and close the file.

Specifying the Blocking Query for Matching

The blocking query defines the query used by the master index to retrieve a candidate selection pool of records that closely match an incoming record and that will be used for probabilistic weighting. The name of the blocking query you specify here must match the name of one of the blocking queries defined in the Candidate Select file (for more information, see **Chapter 3**, **"Candidate Select Configuration"**). You can only specify one blocking query for matching in the master index.

Important: *SeeBeyond advises against modifying the blocking query once your system is in production because it can cause issues with data integrity.*

To specify the blocking query for matching

1 In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **execute-match** element in the **MasterControllerConfig** element.

3 Modify the value of the **query-builder** name to the name of the blocking query you want to use. For example:

```
<execute-match>
    <query-builder name="MY-BLOCKER">
        <option key="key" value="value"/>
    </query-builder>
</execute-match>
```

Important: *Make sure the name you specify exactly matches the name of one of the blocking queries defined in the Candidate Select file.*

4 Set the blocking query options, as described next in **"Setting Blocking Query Options"**.

5 Save and close the file.

Setting Blocking Query Options

Key and value pairs are designed to fine-tune the operation of custom blocking queries. In the default blocking queries defined in the Candidate Select file, key and value pairs are not used. However, if you create a custom blocking query, you can configure the query to use key and value pairs. The key describes the option, and the value specifies the value of the option.

To set blocking query options

1. In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **execute-match** element.
3. To specify a key and value pair, modify the values of the **key** and **value** attributes in the **option** element. For example:

```
<option key="wildcard" value="true"/>
```

4. To delete a key and value pair, remove the **option** element.
5. Save and close the file.

5.4.2 Configuring the Decision Maker

The Decision Maker defines how to handle certain decision points in the matching process, such as how to handle multiple records above the match threshold, whether records originating from the same system can be automatically matched, and weight thresholds. You can customize the Decision Maker by performing any of the following actions:

- **Specifying the Decision Maker Class** on page 64
- **Modifying the OneExactMatch Parameter** on page 65
- **Modifying the SameSystemMatch Parameter** on page 65
- **Specifying the Duplicate Threshold** on page 66
- **Specifying the Match Threshold** on page 66
- **Adding a New Decision Maker Parameter** on page 67

Specifying the Decision Maker Class

If you create your own Decision Maker class, you can specify the new class to be used by changing the value of the **decision-maker-class** element.

To specify the decision maker class

1. In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **DecisionMakerConfig** element.
3. Change the value of the **decision-maker-class** element to the path and name of the new Decision Maker class. For example:

```
<decision-maker-class>com.stc.eindex.decision.impl.MyDecisionMaker
</decision-maker-class>
```

4. Save and close the file.

Modifying the OneExactMatch Parameter

The **OneExactMatch** parameter determines how records above the match threshold are processed. For more information, see **"OneExactMatch" on page 56**.

To enable one exact match processing

1. In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **OneExactMatch** element in the **DecisionMakerConfig** element.
3. Change the value of the **parameter-value** element to **true**. For example:

```
<parameter>
    <parameter-name>OneExactMatch</parameter-name>
    <parameter-type>java.lang.Boolean</parameter-type>
    <parameter-value>true</parameter-value>
</parameter>
```

4. Save and close the file.

To disable one exact match processing

1. In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **OneExactMatch** element in the **DecisionMakerConfig** element.
3. Change value of the **parameter-value** element to **false**. For example:

```
<parameter>
    <parameter-name>OneExactMatch</parameter-name>
    <parameter-type>java.lang.Boolean</parameter-type>
    <parameter-value>false</parameter-value>
</parameter>
```

4. Save and close the file.

Modifying the SameSystemMatch Parameter

The **SameSystemMatch** parameter determines whether two records with local IDs from the same system can be merged automatically. If your local systems contain reliable data, and rarely duplicate their own records, set this parameter to **true**.

To allow same system records to be automatically merged

1. In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **SameSystemMatch** element in the **DecisionMakerConfig** element.
3. Change the value of the **parameter-value** element to **true**. For example:

```
<parameter>
    <parameter-name>SameSystemMatch</parameter-name>
    <parameter-type>java.lang.Boolean</parameter-type>
    <parameter-value>true</parameter-value>
</parameter>
```

4. Save and close the file.

To prevent same system records from being automatically merged

1. In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **SameSystemMatch** element in the **DecisionMakerConfig** element.
3. Change the value of the **parameter-value** element to **false**. For example:

```
<parameter>
    <parameter-name>OneExactMatch</parameter-name>
    <parameter-type>java.lang.Boolean</parameter-type>
    <parameter-value>false</parameter-value>
</parameter>
```

4. Save and close the file.

Specifying the Duplicate Threshold

The duplicate threshold is the lowest matching probability weight at which two records are considered potential duplicates of one another. Any records with lower probability weights are not considered to be possible matches. Any records between the duplicate and match thresholds are flagged as potential duplicates, and must be resolved manually.

To specify the duplicate threshold

1. In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **DuplicateThreshold** element in the **DecisionMakerConfig** element.
3. Change the value of the **parameter-value** element. For example:

```
<parameter>
    <parameter-name>DuplicateThreshold</parameter-name>
    <parameter-type>java.lang.Float</parameter-type>
    <parameter-value>7.9</parameter-value>
</parameter>
```

Note: *This value can be any float value lower than the match threshold but higher than the lowest possible matching probability weight.*

4. Save and close the file.

Specifying the Match Threshold

The **MatchThreshold** parameter specifies the matching probability weight at which two records will be automatically merged, depending on the value of the **OneExactMatch** parameter and the number of records at or above the match threshold.

To specify the match threshold

1. In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **MatchThreshold** element in the **DecisionMakerConfig** element.
3. Change the value of the **parameter-value** element. For example:

```
<parameter>
    <parameter-name>MatchThreshold</parameter-name>
    <parameter-type>java.lang.Float</parameter-type>
    <parameter-value>28.5</parameter-value>
</parameter>
```

Note: *This value can be any float value higher than the duplicate threshold but lower than the highest possible matching probability weight.*

4 Save and close the file.

Adding a New Decision Maker Parameter

New parameters cannot be used with the default Decision Maker class, but if you create your own Decision Maker class, you may need to add new parameters for the new class.

To add a new Decision Maker parameter

1 In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **DecisionMakerConfig** element.

3 Inside the **parameters** tags, but outside any existing **parameter** tags, create a new starting and ending **parameter** tag. For example:

```
<parameters>
    <parameter>
        <parameter-name>OneExactMatch</parameter-name>
        <parameter-type>java.lang.Integer</parameter-type>
        <parameter-value>false</parameter-value>
    </parameter>
<parameter>
    </parameter>
    </parameters>
</parameters>
```

4 Within the new **parameter** tags, add and define the elements described in **Table 10 on page 68**. For example:

```
    <parameters>
        <parameter>
            <parameter-name>OneExactMatch</parameter-name>
            <parameter-type>java.lang.Boolean</parameter-type>
            <parameter-value>false</parameter-value>
        </parameter>
        <parameter>
            <parameter-name>MaxDuplicates</parameter-name>
            <parameter-type>java.lang.Boolean</parameter-type>
            <parameter-value>5</parameter-value>
        </parameter>
    </parameters>
```

5 Save and close the file.

Table 10 Threshold File Parameter Elements

Element	Description
description	A brief description of the parameter. This element is optional.
parameter-name	The name of the parameter.
parameter-type	The type of parameter. Valid values are **java.lang.Long, java.lang.Short**, **java.lang.Byte**, **java.lang.String**, **java.lang.Integer**, **java.lang.Boolean**, **java.lang.Double**, or **java.lang.Float**.
parameter-value	The value of the parameter.

Deleting a Decision Maker Parameter

You cannot delete the default parameters if you are using the default Decision Maker class. However, if you create your own Decision Maker class, some of the default parameters may not work with the new class.

To delete a Decision Maker parameter

1 In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **DecisionMakerConfig** element, and then to the **parameter** element you want to delete.

3 Remove all text between and including the starting and ending **parameter** tag.

For example, to delete the **ExtensiveSearch** parameter in the following sample, you would delete the boldface text.

```
<parameters>
    <parameter>
        <parameter-name>OneExactMatch</parameter-name>
        <parameter-type>java.lang.Boolean</parameter-type>
        <parameter-value>false</parameter-value>
    </parameter>
    <parameter>
        <parameter-name>ExtensiveSearch</parameter-name>
        <parameter-type>java.lang.Boolean</parameter-type>
        <parameter-value>true</parameter-value>
    </parameter>
</parameters>
```

4 To remove all parameters, remove all text between and including the starting and ending **parameters** tag.

5 Save and close the file.

5.4.3 Configuring the EUID Generator

The EUID Generator controls how the enterprise-wide unique identifiers (EUIDs) are created by the master index, including the length of the ID and the checksum value. You can customize the EUID Generator by performing any of the following actions:

- **Specifying the EUID Generator Class** on page 69
- **Specifying the EUID Length** on page 69
- **Specifying a Checksum Length** on page 70
- **Specifying the Chunk Size** on page 70

Specifying the EUID Generator Class

The default EUID generator creates sequential EUIDs based on the value specified for the EUID sequence in the sbyn_seq_table database table. If you create a new EUID generator class to process EUIDs differently, you must specify the name of the new class in the **euid-generator-class** element of the Threshold file.

To specify the EUID Generator Class

1 In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **EuidGeneratorConfig** element.

3 Change the value of the **euid-generator-class** element to the path and name of the new EUID Generator class. For example:

```
<euid-generator-class>com.stc.eindex.idgen.impl.MyEuidGenerator
</euid-generator-class>
```

4 Save and close the file.

Specifying the EUID Length

By default, the length of the EUIDs generated by the master index is 10 characters. You can modify this value if needed. You can also specify a new starting EUID number if needed. This is described in the *eView Studio User's Guide*.

If you increase the length of the EUIDs to longer than 20 characters, or if the EUID length plus the checksum length is longer than 20 characters, you must increase the length of the EUID columns in the script that creates the database tables.

To specify the EUID length

1 In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the first **parameter** element, **IdLength**, in the **EuidGeneratorConfig** element.

3 Change the value of the **parameter-value** element to the desired length of the EUID. For example:

```
<parameter>
    <parameter-name>IdLength</parameter-name>
```

```
        <parameter-type>java.lang.Integer</parameter-type>
        <parameter-value>8</parameter-value>
    </parameter>
```

4 Save and close the file.

Specifying a Checksum Length

A checksum value is a number appended to the end of each EUID that allows systems to validate EUIDs to ensure accurate identification of records throughout the network. For detailed information about checksum values, see **"ChecksumLength" on page 57** and **"Checksum Values and EUID Lengths" on page 57**.

If you specify **0** (zero), checksum values are not used for validation. If the EUID length plus the checksum length is greater than 20, you must increase the length of the EUID columns in the database creation scripts.

To specify a checksum length

1 In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the second **parameter** element, **ChecksumLength**, in the **EuidGeneratorConfig** element.

3 Change the value of the **parameter-value** element to the desired length of the checksum value. For example:

```
<parameter>
    <parameter-name>ChecksumLength</parameter-name>
    <parameter-type>java.lang.Integer</parameter-type>
    <parameter-value>2</parameter-value>
</parameter>
```

4 Save and close the file.

Specifying the Chunk Size

You can specify a number of EUIDs to be allocated at one time to the EUID generator. This allows the generator to assign EUIDs to a number of records without needing to query the sbyn_seq_table database table. For more information about the chunk size, see **"ChunkSize" on page 57**.

To specify the chunk size

1 In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the third **parameter** element, **ChunkSize**, in the **EuidGeneratorConfig** element.

3 Change the value of the **parameter-value** element to the desired chunk size. For example:

```
<parameter>
    <parameter-name>EuidGeneratorConfig</parameter-name>
    <parameter-type>java.lang.Integer</parameter-type>
    <parameter-value>100</parameter-value>
</parameter>
```

4 Save and close the file.

Adding a New EUID Generator Parameter

New parameters cannot be used with the default EUID Generator class, but if you create your own EUID Generator class, you can create additional parameters for the new class.

To add a new EUID Generator parameter

1 In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **EuidGeneratorConfig** element.

3 Inside the **parameters** tags, but outside any existing **parameter** tags, create a new starting and ending **parameter** tag. For example:

```
<parameters>
    <parameter>
        <parameter-name>IdLength</parameter-name>
        <parameter-type>java.lang.Integer</parameter-type>
        <parameter-value>10</parameter-value>
    </parameter>
    <parameter>
    </parameter>
</parameters>
```

4 Within the new **parameter** tags, add the elements described in **Table 10 on page 68**. For example:

```
<parameters>
    <parameter>
        <parameter-name>IdLength</parameter-name>
        <parameter-type>java.lang.Integer</parameter-type>
        <parameter-value>10</parameter-value>
    </parameter>
    <parameter>
        <parameter-name>StartNumber</parameter-name>
        <parameter-type>java.lang.Integer</parameter-type>
        <parameter-value>1000000001</parameter-value>
    </parameter>
</parameters>
```

5 Save and close the file.

Deleting an EUID Generator Parameter

You cannot delete the default parameters if you are using the default EUID Generator class. If you create your own EUID Generator class, some of the default parameters may not work with the new class.

To delete an EUID Generator parameter

1 In the Project Explorer pane of the Enterprise Designer, double-click **Threshold** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **EuidGeneratorConfig** element, and then to the **parameter** element you want to delete.

3 Remove all text between and including the starting and ending **parameter** tag.

For example, to remove the **StartNumber** parameter in the sample below, delete all boldface text.

```
<parameters>
    <parameter>
        <parameter-name>IdLength</parameter-name>
        <parameter-type>java.lang.Integer</parameter-type>
        <parameter-value>10</parameter-value>
    </parameter>
    <parameter>
        <parameter-name>StartNumber</parameter-name>
        <parameter-type>java.lang.Integer</parameter-type>
        <parameter-value>1000000001</parameter-value>
    </parameter>
</parameters>
```

4 To remove all parameters, remove all text between and including the starting and ending **parameters** tag.

5 Save and close the file.

Chapter 6

Match Field Configuration

The Match Field file configures the Matching Service, which contains the matching and standardization engines used in the match process, as well as the phonetic encoders used for phoneticizing names. This chapter describes the components of the Matching Service and the structure of the Match Field file, and provides instructions for customizing the Match Field file.

6.1 About Match Field Configuration

The Match Field file specifies the match and standardization engines for the master index and includes special standardization, matching, and weighting logic used by the engines. It also defines the strategy for identifying unique records and finding the best matches in the master index database. For optimization, the Match Field components are configurable, allowing you to choose the strategy that best fits your requirements or to implement your own custom components.

6.2 Matching Service Components

The Matching Service is configured by the Match Field file, which defines the configurable properties for standardizing data and matching records. These processes are highly configurable in eView, allowing you to design and develop the match strategy that best suits your processing requirements. The following components make up the Matching Service.

- **Standardization Configuration** on page 73
- **Matching Configuration** on page 74
- **MEFA Configuration** on page 74
- **Phonetic Encoders** on page 75

6.2.1 Standardization Configuration

Standardization of incoming data applies three functions to the data processed by the master index: reformatting (or parsing), normalization, and phonetic encoding. These functions help prepare data for matching and searching. Some fields might require all three steps, some just normalization and phonetic conversion, and other data might only need phonetic encoding. You can specify which fields require any of these steps in the standardization configuration section of the Match Field file. In addition, you can specify the nationality of the data being standardized by the SeeBeyond Match Engine.

Reformatting

If incoming records contain data that is not formatted properly, it must be reformatted before it can be normalized. One good example of this is freeform text address fields. If you are matching or searching on street addresses that are contained in one or more freeform text fields (that is, the street address is contained in one field, apartment number in another, and so on), that field must be parsed into its individual components (house number, street name, street type, and so on) before the data can be normalized.

Normalization

When you normalize data, the data is converted into a standard form. A common use for normalization is to convert nicknames into their standard names, such as converting “Rich” to “Richard” or “Meg” to “Margaret”. Another example is normalizing street address components. For example, “Dr.” or “Drv” in a street address might be normalized to “Drive”. Normalized values are obtained from look-up tables.

Phonetic Encoding

Once data has gone through any necessary reformatting and normalization, it can be phonetically encoded. Phonetic values are generally used in blocking queries in order to obtain all possible matches to an incoming record. They are also used to perform searches from the EDM that allow for misspellings and typographic errors. Typically, first names use Soundex encoding and last names and street names use NYSIIS encoding.

6.2.2 Matching Configuration

The **MatchingConfig** section of the Match Field file allows you to define the data fields that are sent to the match engine (called the *match string*). Probabilistic weighting is performed only against the fields you specify as the match columns. You can specify any field in the object structure as a match column as long as the match engine is configured to use all fields specified. The configuration of this section of the Match Field file is specific to the match engine you are using and the types of fields on which you are matching. For more information about how the matching should be configured for each match engine, see the implementation guide for the appropriate match engine type (*Implementing the SeeBeyond Match Engine with eView Studio* or *Implementing Ascential INTEGRITY with eView Studio*).

6.2.3 MEFA Configuration

The **MEFAConfig** section specifies the Java classes to be used by the following Matching Service components.

- Match and Standardization Engines
- Block Picker and Pass Controller

The match and standardization engines control the processes of standardizing data and generating matching probability weights between records. The block picker and pass controller define how the blocking query is executed during the match process.

Match and Standardization Engines

eView provides the ability to use the standardization and match engines that best suit your indexing requirements. You can configure the master index to use one of the standard engines from SeeBeyond or Ascential, or you can configure the index to use a customized engine of your choice.

These engines perform two functions:

- Standardize data to a common format
- Calculate the likelihood that two objects match

The engines are called during match processing, when the index retrieves the best matches during a weighted search from the EDM, or when the master index checks for duplicate records during an insert or update from the EDM or an external system.

Block Picker and Pass Controller

By default, the matching process is executed in multiple stages. Each configured block that defines query parameters is executed and evaluated separately (each query execution and evaluation is referred to as a “match pass”). After a block is evaluated, the pass controller determines whether the results found are sufficient or matching should continue by performing another match pass.

The block picker chooses the block definition to use for each match pass. Block definitions define the criteria for each query that checks the database for a subset of the records to be used for matching. The block picker has access to the match results from previous match passes, as well as lists of applicable block definitions that have been executed and of those that have not been executed.

6.2.4 Phonetic Encoders

eView provides extensible phonetic encoding capabilities, which are typically used to retrieve records with similar fields from the database for matching. By default, the Soundex and NYSIIS phonetic encoders are defined to be used in the master index implementation. Typically, Soundex is used to encode first names and NYSIIS to encode last names. When using the SeeBeyond Match Engine, you can specify different types of phonetic encoders (but you must also create a Java class to implement the encoder). When you specify the fields in the standardization configuration to be phonetically encoded, you can select one of the encoders defined in the phonetic encoders section.

6.3 Sample Standardization and Matching Sequence

The following steps illustrate one possible processing sequence that occurs when data is received from an external system and processed by the master index.

1. A record is received from an external system.
   - The local ID does not yet exist in the master index; initiate the standardization and matching process.
2. Standardize the record to a common format.
   - Standardize freeform text.
   - Normalize single fields.
   - Phonetically encode fields that are commonly misspelled or spelled in different ways.
3. Match the record against entries in the database.
   - Use the selected blocking query (specified in the Threshold file) to use to retrieve a block of records that might match the new record.
   - Build and execute the query according to the input record.
   - Calculate match scores comparing the incoming record against existing records (this is done by the match engine).
   - Determine whether to repeat the matching process with another block of records, based on the **MEFAConfig** element in the Match Field file.
4. Return match scores for further processing.
   - Determine whether to add the system record to an existing EUID record or to insert the system record as a new EUID record (based on the parameters defined in the **DecisionMaker** element of the Threshold file).

6.4 The Match Field File

The properties for the match and standardization process are defined in the Match Field file in XML format. Some of the information entered into the default configuration file is taken from the eView Wizard, but the file may require additional customization in order to meet your data processing needs.

6.4.1 Modifying the Match Field File

You can modify the Match Field file at any time, but modifying the file is not recommended once you move to production because this file defines how records are processed and data integrity is maintained. You must regenerate the Project and reactivate the deployment after making any changes to this file. You can either apply the changes immediately to the Logical Host, or restart the host to pick up the changes.

This may cause weighting and standardization to be handled differently, causing unexpected match weight results.

Before making any changes to this file, make sure you are familiar with the procedures described under **"Modifying Configuration Files" on page 24**. The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes.

6.4.2 Match Field File Structure

The Match Field file is written in XML and is automatically generated by the eView Wizard. This file can be modified using the Enterprise Designer XML editor, which is a standard NetBeans editor. This section describes the structure of the XML file and general requirements and constraints. There are five primary components of the Match Field file. The first component, XML File Information, is standard across all implementations.

- XML File Information
- Standardization Configuration
- Matching Configuration
- MEFA Configuration
- Phonetic Encoder Configuration

XML File Information

The first lines in the Match Field file represent information about the schema instance and namespace. You should not need to modify these components.

Standardization Configuration

The **StandardizationConfig** section of the Match Field file consists of several structures that define standardization rules for a set of fields. The object whose fields you specify for standardization is defined by the **standardize-system-object** element, and named by the **system-object-name** element, as shown below.

```
<standardize-system-object>
    <system-object-name>Person</system-object-name>
```

Each set of **standardize-system-object** tags includes one standardization structure. The objects you specify must be defined in the Object Definition file. The object can be the parent object, which allows you to specify any field in any object for standardization, or you can create multiple standardization structures and specify a different object for each structure.

Each standardization structure contains three primary elements: **structures-to-normalize**, **free-form-texts-to-standardize**, and **phoneticize-fields**. These elements are all required for a **standardize-system-object** element; however any of these elements can be empty.

structures-to-normalize

This section defines the fields that require normalization (but not parsing or reformatting) before being processed by the match engine, along with the nationality of the data in those fields (for the SeeBeyond Match Engine only). This section may contain one or more **group** elements, each of which defines source and target fields for one normalization unit. Each group contains at least one **unnormalized-source-fields** element to define the source fields for normalization (**source-mapping**), and at least one **normalization-targets** element to map the source fields to target fields (**target-mapping**) in the system object.

The elements in a normalization structure are described in **"Defining Normalization Structures" on page 83**. Below is a sample source and target field mapping in a normalization structure.

```
<structures-to-normalize>
     <group standardization-type="PersonName"domain-selector=
          "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
          <unnormalized-source-fields>
               <source-mapping>
                    <unnormalized-source-field-name>Person.FirstName
                    </unnormalized-source-field-name>
                    <standardized-object-field-id>FirstName
                    </standardized-object-field-id>
               </source-mapping>
          </unnormalized-source-fields>
          <normalization-targets>
               <target-mapping>
                    <standardized-object-field-id>FirstName
                    </standardized-object-field-id>
                    <standardized-target-field-name>Person.FirstName_Std
                    </standardized-target-field-name>
               </target-mapping>
          </normalization-targets>
     </group>
```

free-form-texts-to-standardize

This section defines the fields that require parsing or reformatting and, optionally, normalization. It also defines the nationality of the data to standardize when using the SeeBeyond Match Engine. This section may contain one or more **group** elements, each of which defines source and target fields for one standardization unit. Each group contains at least one **unstandardized-source-fields** element, which defines the source fields for standardization, and at least one **standardization-targets** element to map the source fields to target fields (**target-mapping**) in the system object.

Note: *If you define more than one source field for mapping (**unstandardized-source-field-name**) in the same standardization structure, the fields are concatenated during standardization, with a pipe (|) between lines for the SeeBeyond Match Engine or a space character between lines for INTEGRITY.*

The elements in a standardization structure are described in **"Defining Standardization Structures" on page 89**. Below is a sample source and target field mapping in a simplified address standardization structure.

```
<group standardization-type="Address"
     domain-selector="com.stc.eindex.matching.impl.SingleDomainSelectorUK">
     <unstandardized-source-fields>
          <unstandardized-source-field-name>Person.Address[*].Address1
          </unstandardized-source-field-name>
     </unstandardized-source-fields>
     <standardization-targets>
          <target-mapping>
           <standardized-object-field-id>HouseNumber
           </standardized-object-field-id>
           <standardized-target-field-name>Person.Address[*].Address1HouseNo
           </standardized-target-field-name>
          <target-mapping>
           <standardized-object-field-id>OrigStreetName
           </standardized-object-field-id>
           <standardized-target-field-name>Person.Address[*].Address1StName
           </standardized-target-field-name>
          </target-mapping>
          <target-mapping>
           <standardized-object-field-id>StreetNameSufType
           </standardized-object-field-id>
           <standardized-target-field-name>Person.Address[*].Address1StType
           </standardized-target-field-name>
          </target-mapping>
     </standardization-targets>
</group>
```

phoneticize-fields

This section defines the fields that must be converted into phonetic form. This section may contain one or more **phoneticize-field** elements, each of which defines the source field, destination field, and the type of phonetic encoder to be used for one field. Each **phoneticize-field** element contains a source field name, a destination field name, an encoder type, and optionally, an ID to identify the destination field to the match engine.

The elements in the **phoneticize-fields** section are described in **"Defining Phonetic Encoding" on page 95**. Below is a sample of the **phoneticize-fields** section.

```
<phoneticize-fields>
     <phoneticize-field>
          <unphoneticized-source-field-name>Person.FirstName_Std
          </unphoneticized-source-field-name>
          <phoneticized-target-field-name>Person.FirstName_Phon
          </phoneticized-target-field-name>
          <encoding-type>Soundex</encoding-type>
     </phoneticize-field>
     <phoneticize-field>
          <unphoneticized-source-field-name>Person.LastName_Std
          </unphoneticized-source-field-name>
          <phoneticized-target-field-name>Person.LastName_Phon
          </phoneticized-target-field-name>
          <encoding-type>NYSIIS</encoding-type>
     </phoneticize-field>
```

Matching Configuration

The **MatchingConfig** section of the Match Field file defines the fields that are included in the data string sent to the match engine and against which weighting is performed. This is called the "match string" (though it might not always consist of a string). The match string is identified in a **match-system-object** element and is named by an **object-name** element.

The **match-system-object** element includes a **match-columns** element, which contains a list of all the fields in the match string. Each match field is identified by a **match-column** element, which defines the field using the following elements.

- **column-name**
  The fully qualified field name of the field to be included in the match string.
- **match-type**
  The type of matching performed on the specified field. This element is unique to each match engine. For more information about this element for the match engine you are using, see the appropriate match engine implementation guide.
- **match-order**
  The order in which the field should appear in the match string. This value is an integer, and this element is optional. If no order is specified, matching is performed in the order in which the fields are listed.

The elements of the **MatchingConfig** element are described in **"Defining the Match Object" on page 97**. Below is a short sample of a match string containing only the first and last names.

```
<match-system-object>
     <object-name>Person</object-name>
     <match-columns>
          <match-column>
           <column-name>Enterprise.SystemSBR.Person.FirstName</column-name>
           <match-type>FirstName</match-type>
           <match-order>1</match-type>
          </match-column>
          <match-column>
           <column-name>Enterprise.SystemSBR.Person.LastName</column-name>
           <match-type>LastName</match-type>
           <match-order>2</match-type>
          </match-column>
     </match-columns>
</match-system-object>
```

MEFA Configuration

The **MEFAConfig** section of the Match Field file allows you to specify the Java class for each element listed below. The fully qualified name for each Java class is specified by the corresponding **class-name** element. **MEFAConfig** can only contain one instance of each element, and each element is required. Sample configurations for each match engine follow the descriptions.

- **block-picker**
  The **block-picker** element identifies the Java class that chooses which block of criteria defined for the blocking query to use for each match pass.
- **pass-controller**
  The **pass-controller** element determines whether the blocking query should continue performing match passes after each match pass is complete.
- **standardizer-api**
  The **standardizer-api** element identifies the standardization engine to use. eView provides default classes for the SeeBeyond Match Engine and Ascential INTEGRITY.
- **standardizer-config**
  The **standardizer-config** element provides the standardization engine with configuration information. eView provides default classes for the SeeBeyond Match Engine and Ascential INTEGRITY.

- **matcher-api**
  The **matcher-api** element identifies the match engine to use. eView provides default classes for the SeeBeyond Match Engine and Ascential INTEGRITY.
- **matcher-config**
  The **matcher-config** element provides the match engine with configuration information. eView provides default classes for the SeeBeyond Match Engine and Ascential INTEGRITY.

Following is a sample MEFA configuration for the SeeBeyond Match Engine.

```
<block-picker>
     <class-name>com.stc.eindex.matching.impl.PickAllBlocksAtOnce
     </class-name>
</block-picker>
<pass-controller>
     <class-name>com.stc.eindex.matching.impl.PassAllBlocks</class-name>
</pass-controller>
<standardizer-api>
     <class-name>com.stc.eindex.matching.adapter.SbmeStandardizerAdapter
     </class-name>
</standardizer-api>
<standardizer-config>
     <class-name>com.stc.eindex.matching.adapter.SbmeStandardizerAdapterConfig
     </class-name>
</standardizer-config>
<matcher-api>
     <class-name>com.stc.eindex.matching.adapter.SbmeMatcherAdapter
     </class-name>
</matcher-api>
<matcher-config>
     <class-name>com.stc.eindex.matching.adapter.SbmeMatcherAdapterConfig
     </class-name>
</matcher-config>
```

Following is a sample MEFA configuration for INTEGRITY.

```
<block-picker>
     <class-name>com.stc.eindex.matching.impl.PickAllBlocksAtOnce</class-name>
</block-picker>
<pass-controller>
     <class-name>com.stc.eindex.matching.impl.PassAllBlocks</class-name>
</pass-controller>
<standardizer-api>
     <class-name>com.stc.eindex.matching.adapter.IntegrityStandardizerAdapter
     </class-name>
</standardizer-api>
<standardizer-config>
     <class-name>
          com.stc.eindex.matching.adapter.IntegrityStandardizerAdapterConfig
     </class-name>
</standardizer-config>
<matcher-api>
     <class-name>com.stc.eindex.matching.adapter.IntegrityMatcherAdapter
     </class-name>
</matcher-api>
<matcher-config>
     <class-name>com.stc.eindex.matching.adapter.IntegrityMatcherAdapterConfig
     </class-name>
</matcher-config>
```

Phonetic Encoder Configuration

The **PhoneticEncodersConfig** section of the Match Field file defines the types of phonetic encoders available to the implementation. Two encoders, NYSIIS and Soundex, are defined by default. Each encoder is specified by an encoder element, which contains one **encoding-type** element and one **encoder-implementation-class** element. The **encoding-type** element is the name of the encoder, and the **encoder-**

implementation-class element is the fully qualified name of the Java class for the encoder. Below is the default **PhoneticEncodersConfig** section from the Match Field file.

```
<PhoneticEncodersConfig module-name="PhoneticEncoders" parser-
class="com.stc.eindex.configurator.impl.PhoneticEncodersConfig">
     <encoder>
      <encoding-type>NYSIIS</encoding-type>
      <encoder-implementation-class>com.stc.eindex.phonetic.impl.Nysiis
      </encoder-implementation-class>
     </encoder>
     <encoder>
      <encoding-type>Soundex</encoding-type>
      <encoder-implementation-class>com.stc.eindex.phonetic.impl.Soundex
      </encoder-implementation-class>
     </encoder>
</PhoneticEncodersConfig>
```

6.5 Customizing the Match Field File

Configuring the match fields is a complicated task and requires a thorough analysis of your existing data in order to determine the best configuration for your processing needs. The Match Field file consists of four sections:

- **Standardization Configuration** on page 82
- **Matching Configuration** on page 97
- **MEFA Configuration** on page 102
- **Phonetic Encoding** on page 105

All of these actions require that you use the Enterprise Designer XML editor. For more information about this editor, see **“Using the eView Editors” on page 21**. Make sure you are familiar with the procedures described under **“Modifying Configuration Files” on page 24** for checking the file in and out, saving changes, and validating the file.

Important: *Much of the configuration of this file is dependent upon the match engine you use in your implementation. See the appropriate match engine implementation guide for additional configuration information specific to the match engine in use.*

6.6 Standardization Configuration

Standardization consists of three steps: reformatting, normalization, and phonetic encoding. Depending on the incoming data, fields may require parsing and normalization (such as a freeform text address field) or may already be parsed correctly and only require the normalization step (such as when the first and last names appear in separate fields). Configuring the standardization process can be divided into the following three tasks:

- **Defining Normalization Structures on page 83**

- **Defining Standardization Structures on page 89**
- **Defining Phonetic Encoding on page 95**

6.6.1 Defining Normalization Structures

If any fields used for searching or matching are already in the correct format but require normalization, those fields must be specified in the **structures-to-normalize** section of the Match Field file. Once the specified fields are normalized, phonetic versions of the fields are created using rules defined in the **phoneticize-fields** section. To specify that certain data be normalized, you must specify the objects containing the fields to normalize, the source fields to normalize, and the destination fields for the normalized data.

eView uses the standardization engine defined in **"Configuring the Standardization Engine" on page 104** to determine how the data is normalized. You can customize the normalization process by performing any of the following actions:

- **Defining the Object for the Normalization Structure** on page 83
- **Creating a Normalization Structure** on page 84
- **Specifying Source Fields to Normalize** on page 85
- **Mapping Normalized Data to Destination Fields** on page 86
- **Modifying a Normalization Structure** on page 87
- **Deleting a Normalization Structure** on page 88

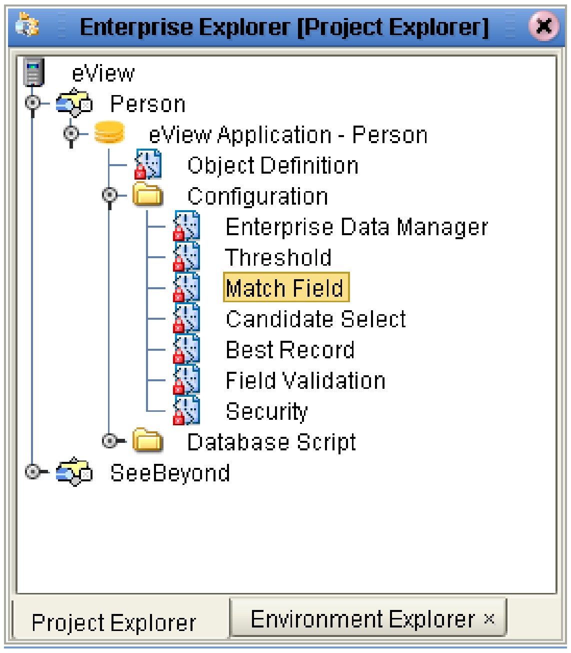
Defining the Object for the Normalization Structure

You can specify any of the objects defined in the Object Definition file for normalization if fields in that object are supported by the match engine in use. By default, the parent object is defined, so any fields in the parent or child objects can be defined for normalization.

To define the object

1. In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

Figure 5 Match Field Node



2 Scroll to the **standardize-system-object** element, and then to the **system-object-name** element

3 Modify the name of the system object. For example:

```
<standardize-system-object>
    <system-object-name>Person</system-object-name>
```

Note: *The name you specify must correspond to the object's name in the Object Definition file.*

4 Save and close the file.

Creating a Normalization Structure

Before you can specify the fields to normalize, you must specify the normalization structure to which those fields belong. Each normalization structure falls within a **group** element. The type of normalization to perform on each group is defined by a **standardization-type** attribute and the nationality of the data being normalized is defined by a **domain-selector** attribute. These attributes are specific to the match engine being used. For more information, see “Match and Standardization Types” in Appendix B of the *eView Studio User's Guide*.

To create a normalization structure

1 In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **structures-to-normalize** element in the **StandardizationConfig** element.

3 Create and name a new **group** element in the **structures-to-normalize** element.

Make sure the new element falls within the **structures-to-normalize** element, but outside any existing **group** tags.

4 In the new **group** element, define the attributes described in **Table 11 on page 85**. For example:

```
<structures-to-normalize>
    <group standardization-type="PersonName"domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
        <unnormalized-source-fields>
        ...
    </group>
    <group standardization-type="PersonName" domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
    </group>
</structures-to-normalize>
```

5 Specify the source fields to normalize, as described in **"Specifying Source Fields to Normalize"**.

6 Map the normalized data to destination fields, as described in **"Mapping Normalized Data to Destination Fields"**.

7 Save and close the file.

Table 11 Normalization Structure Group Attributes

Attribute	Description
standardization-type	The type of standardization to perform on the source fields. This is specific to the match engine being used and the type of data being processed. For more information, see the appropriate match engine implementation guide.
domain-selector	The Java class used by the SeeBeyond Match Engine to determine the nationality of the data being processed. This is not used for INTEGRITY implementations. Currently, the following two classes can be specified, the first to standardize United States data and the second to standardize United Kingdom data. If no selector is specified, the default is US. ▪ com.stc.eindex.matching.impl.SingleDomainSelectorUS ▪ com.stc.eindex.matching.impl.SingleDomainSelectorUK

Specifying Source Fields to Normalize

In order for the standardization engine to know which data fields to normalize, you must specify the fields that you want converted to their normalized form. Typical fields for normalization might include first and last name for person objects or business names for company objects.

To specify source fields to normalize

1 In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **structures-to-normalize** element in the **StandardizationConfig** element, and then to the **group** element for which you are specifying source fields.

3. Create a new **unnormalized-source-fields** element in the **group** element.

```
<structures-to-normalize>
    <group standardization-type="PersonName" domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
        <unnormalized-source-fields>
        </unnormalized-source-fields>
    </group>
</structures-to-normalize>
```

4. Create a **source-mapping** element in the new **unnormalized-source-fields** element.
5. Define the elements described in Table 12 in the new element. For example:

```
<group standardization-type="PersonName" domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
    <unnormalized-source-fields>
        <source-mapping>
            <unnormalized-source-field-name>Person.Alias[*].LName
            </unnormalized-source-field-name>
            <standardized-object-field-id>LastName
            </standardized-object-field-id>
        </source-mapping>
    </unnormalized-source-fields>
</group>
```

6. Map the normalized data to destination fields, as described in **"Mapping Normalized Data to Destination Fields"**.
7. Save and close the file.

Table 12 Normalization Structure Source Mapping Elements

Element	Description
unnormalized-source-field-name	The ePath of the source field to normalize in the system object (for example, **Person.FirstName**).
standardized-object-field-id	An identification code that identifies the field to normalize to the standardization engine. This ID is specific to the standardization engine in use and must correspond to a field ID defined by that engine. For more information, see the appropriate match engine implementation guide.

Mapping Normalized Data to Destination Fields

Once you specify the source fields to be normalized, you must define the fields in which the modified data will be stored. There is a one to one correspondence between the source fields defined earlier and the target fields defined here.

To map normalized data to destination fields

1. In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **structures-to-normalize** element in the **StandardizationConfig** element.

3 Create a new **normalization-targets** element under the **unnormalized-source-fields** element that defines the field to map. For example:

```
<group standardization-type="PersonName" domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
    <unnormalized-source-fields>
        <source-mapping>
            <unnormalized-source-field-name>Person.Alias[*].LName
            </unnormalized-source-field-name>
            <standardized-object-field-id>LastName
            </standardized-object-field-id>
        </source-mapping>
    </unnormalized-source-fields>
    <normalization-targets>
    </normalization-targets>
</group>
```

4 Create a **target-mapping** element in the new **normalization-targets** element.

5 Define the elements described in Table 13 in the new tags. For example:

```
<group standardization-type="PersonName" domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
    <unnormalized-source-fields>
        <source-mapping>
            <unnormalized-source-field-name>Person.Alias[*].LName
            </unnormalized-source-field-name>
            <standardized-object-field-id>LastName
            </standardized-object-field-id>
        </source-mapping>
    </unnormalized-source-fields>
    <normalization-targets>
        <target-mapping>
            <standardized-object-field-id>LastName
            </standardized-object-field-id>
            <standardized-target-field-name>Person.Alias[*].StdLName
            </standardized-target-field-name>
        </target-mapping>
    </normalization-targets>
</group>
```

6 Save and close the file.

Table 13 Normalization Structure Target Mapping Elements

Element	Description
standardized-object-field-id	An identification code that identifies the normalized field to the standardization engine. This is specific to the standardization engine in use and must correspond to a field ID defined by that engine. For more information, see the appropriate match engine implementation guide.
standardized-target-field-name	The ePath of the target field in which the normalized value is saved in the system object (for example, **Person.Alias[*].StdLastName**).

Modifying a Normalization Structure

After a normalization structure is defined, you can modify the elements of that structure.

To modify a normalization structure

1. In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **structures-to-normalize** element in the **StandardizationConfig** element.
3. To modify the normalization type, change the value of the **standardization-type** attribute.
4. To change the national domain (for SeeBeyond Match Engine implementations), change the value of the **domain-selector** element as described in **Table 11 on page 85**.
5. To modify an existing source field, scroll to the **unnormalized-source-fields** element for the appropriate **group** element, and then change the value of any elements described in **Table 12 on page 86**.
6. To modify an existing destination field, scroll to the **normalization-targets** element in the appropriate **group** element, and then change the value of any elements described in **Table 13 on page 87**.
7. Save and close the file.

Deleting a Normalization Structure

If a defined normalization structure is not required, you can delete the normalization structure from the standardization configuration. If no data requires normalization, you can delete all normalization structures.

To delete a normalization structure

1. In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **structures-to-normalize** element in the **StandardizationConfig** element.
3. Do either of the following:
    - To delete an existing normalization structure, delete all text between and including the start and end **group** tags that define the structure.

      Using the following sample, to delete the USLNAME structure, delete the boldface text.

```
<structures-to-normalize>
    <group standardization-type="USFNAME">
        ...
    </group>
    <group standardization-type="USLNAME">
        ...
    </group>
</structures-to-normalize>
```

    - To specify that no objects require normalization, delete all text between, but not including, the starting and ending **structures-to-normalize** tags.

4 Save and close the file.

6.6.2 Defining Standardization Structures

If any of the fields against which searching or matching is performed are entered in freeform text format, those fields must be standardized before being sent to the match engine. Standardization is the process of parsing, or reformatting, data and then normalizing the reformatted data. After fields are parsed and normalized, phonetic versions of the parsed fields can be created according to rules defined in the **phoneticize-fields** element of the Match Field file. The fields to be standardized must be specified in the **free-form-texts-to-standardize** element of the Match Field file.

To specify that certain data be parsed and then normalized before being processed by the match engine, you need to specify the objects containing the fields to reformat, the source fields to reformat, and the destination fields for the reformatted data. eView uses the standardization engine defined in **"Configuring the Standardization Engine" on page 104** to determine how the data is reformatted. You can customize this process by performing any of the following actions:

- **Creating the Standardization Structure** on page 89
- **Specifying Source Fields to Standardize** on page 90
- **Mapping Standardized Data to Destination Fields** on page 91
- **Modifying a Standardization Structure** on page 92
- **Deleting a Standardization Structure** on page 93
- **Deleting a Source Field from a Standardization Structure** on page 93
- **Deleting a Destination Field for Standardized Data** on page 94

Creating the Standardization Structure

Before you can specify the fields to standardize, you must specify a standardization structure for those fields. Each standardization structure falls between **group** tags. The type of standardization to perform on each group is defined by the **standardization-type** attribute and the nationality of the data being standardized is defined by the **domain-selector** attribute (for the SeeBeyond Match Engine only). For each standardization structure, you can specify more than one field, but they must use the same standardization type. The source fields in one standardization structure are concatenated to determine the parsed field values.

The type of object you specify must correspond to a standardization type defined by the match engine being used. For more information, see "Match and Standardization Types" in Appendix B of the *eView Studio User's Guide*.

To create the standardization structure

1 In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **free-form-texts-to-standardize** element in the **StandardizationConfig** element.

3 Create a new **group** element in the **free-form-texts-to-standardize** element, and then define the attributes described in **Table 14 on page 90**.

Make sure the new element falls within the **free-form-texts-to-standardize** element, but outside any existing **group** tags. For example:

```
<free-form-texts-to-standardize>
    <group standardization-type="BusinessName" domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
        ...
    </group>
    <group standardization-type="Address" domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
    </group>
</free-form-texts-to-standardize>
```

4 Specify the source fields to standardize, as described in **"Specifying Source Fields to Standardize"**.

5 Specify the destination fields for the standardized data as described in **"Mapping Standardized Data to Destination Fields"**.

6 Save and close the file.

Table 14 Standardization Structure Group Attributes

Attribute	Description
standardization-type	The type of standardization to perform on the source fields. This is specific to the match engine being used and the type of data being processed. For more information, see the appropriate match engine implementation guide.
domain-selector	The Java class used by the SeeBeyond Match Engine to determine the nationality of the data being processed. This is not used for INTEGRITY implementations. Currently, the following two classes can be specified, the first to standardize United States data and the second to standardize United Kingdom data. If neither is specified, the default is US. ▪ com.stc.eindex.matching.impl.SingleDomainSelectorUS ▪ com.stc.eindex.matching.impl.SingleDomainSelectorUK

Specifying Source Fields to Standardize

In order for the standardization engine to know which data fields to standardize, you must specify the fields containing the text that must be reformatted and normalized.

To specify source fields to standardize

1 In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **free-form-texts-to-standardize** element in the **StandardizationConfig** element.

3 If it does not currently exist, create an **unstandardized-source-fields** element in the appropriate **group** element (each group element can only include one **unstandardized-source-fields** element).

```
<group standardization-type="Address" domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
    <unstandardized-source-fields>
    </unstandardized-source-fields>
</group>
```

4. For each field standardized by the specified standardization type, create and name a new **unstandardized-source-field-name** element in the new **unstandardized-source-fields** element. For example:

```
<group standardization-type="Address" domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
    <unstandardized-source-fields>
        <unstandardized-source-field-name>Person.Address[*].Address1
        </unstandardized-source-field-name>
        <unstandardized-source-field-name>Person.Address[*].Address2
        </unstandardized-source-field-name>
    </unstandardized-source-fields>
```

Note: *If more than one source field is defined, the fields are concatenated prior to standardization (with a pipe (|) between them for the SeeBeyond Match Engine or a space character between them for INTEGRITY). If you want the fields to be processed separately, you must create two standardization structures.*

5. Specify the destination fields for the standardized data as described in **"Mapping Standardized Data to Destination Fields"**.
6. Save and close the file.

Mapping Standardized Data to Destination Fields

Once you specify the source fields to be standardized, you need to define the fields in which the modified data is stored. Each field is identified by a field ID, and these IDs must correspond to a field ID defined by the match engine in use. There may be more destination than source fields because the data may be parsed into several components during standardization.

To map parsed data to destination fields

1. In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **free-form-texts-to-standardize** element in the **StandardizationConfig** element.
3. In the **group** element for which destination fields need to be defined, create a **standardization-targets** element after the **unstandardized-source-fields** element.

```
<group standardization-type="Address" domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
    <unstandardized-source-fields>
        <unstandardized-source-field-name>Person.Address[*].AddressLine1
        </unstandardized-source-field-name>
        <unstandardized-source-field-name>Person.Address[*].AddressLine2
        </unstandardized-source-field-name>
    </unstandardized-source-fields>
    <standardization-targets>
    </standardization-targets>
```

4 In the new element, create a **target-mapping** element for each destination field, and then define the elements described in Table 15. For example:

```
<group standardization-type="Address" domain-selector=
         "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
     <unstandardized-source-fields>
          <unstandardized-source-field-name>Person.Address[*].AddressLine1
          </unstandardized-source-field-name>
          <unstandardized-source-field-name>Person.Address[*].AddressLine2
          </unstandardized-source-field-name>
     </unstandardized-source-fields>
     <standardization-targets>
          <target-mapping>
               <standardized-object-field-id>HouseNumber
               </standardized-object-field-id>
               <standardized-target-field-name>Person.Address[*].HouseNumber
               </standardized-target-field-name>
          </target-mapping>
          <target-mapping>
               <standardized-object-field-id>OrigStreetName
               </standardized-object-field-id>
               <standardized-target-field-name>Person.Address[*].StreetName
               </standardized-target-field-name>
          </target-mapping>
     </standardization-targets>
</group>
```

5 Save and close the file.

Table 15 Standardization Structure Target Mapping Elements

Element	Description
standardized-object-field-id	An abbreviation that identifies the destination field to the configured standardization engine. This must correspond to a field ID defined by the match engine being used. For more information, see the appropriate match engine implementation guide.
standardized-target-field-name	The ePath of the destination field in the object where the standardized value will be saved (for example, **Person.Address[*].StreetName**).

Modifying a Standardization Structure

After a standardization structure is created in the Match Field file, you can modify the elements of that structure.

To modify a standardization structure

1 In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **structures-to-normalize** element in the **StandardizationConfig** element.

3 To modify the standardization type, change the value of the **standardization-type** attribute.

4 To change the national domain (for SeeBeyond Match Engine implementations), change the value of the **domain-selector** element as described in **Table 11 on page 85**.

5 To modify an existing source field, scroll to the appropriate **group** element, and then change the value of the **unstandardized-source-field-name** element to the ePath of the new field.

6 To modify an existing destination field, scroll to the **target-mapping** element in the **standardization-targets** section, and then change the value of any elements described in Table 15.

7 Save and close the file.

Deleting a Standardization Structure

If a defined standardization structure is not required, you can delete the structure from the standardization configuration. If no data requires standardization, you can delete all standardization structures, but you must retain the **free-form-texts-to-standardize** element.

To delete a standardization structure

1 In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **free-form-texts-to-standardize** element in the **StandardizationConfig** element.

3 Do either of the following:

- To delete an existing standardization structure, delete all text between and including the start and end **group** tags that define the structure.
  Using the example below, to delete the Address object, delete all boldface text.

```
<free-form-texts-to-standardize>
    <group standardization-type="BusinessName" domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
        ...
    </group>
    <group standardization-type="Address" domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
        ...
    </group>
</free-form-texts-to-standardize>
```

- To specify that no fields require standardization, delete all text between, but not including, the starting and ending **free-form-texts-to-standardize** tags. This deletes all standardization structures.

4 Save and close the file.

Deleting a Source Field from a Standardization Structure

If source fields defined in a standardization structure do not require standardization, you can delete the field definitions from the structure.

To delete a source field from a standardization structure

1 In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **free-form-texts-to-standardize** element in the **StandardizationConfig** element.

3 Delete all text between and including the start and end **unstandardized-source-field-name** tags that define the field.

Using the example below, to delete the Address2 field, delete the boldface text.

```
<group standardization-type="Address" domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
    <unstandardized-source-fields>
        <unstandardized-source-field-name>Person.Address[*].Address1
        </unstandardized-source-field-name>
        <unstandardized-source-field-name>Person.Address[*].Address2
        </unstandardized-source-field-name>
    </unstandardized-source-fields>
    ...
</group>
```

Note: *If no fields require standardization in a defined standardization structure, delete the entire structure as described previously in* ***"Deleting a Standardization Structure"****.*

4 Save and close the file.

Deleting a Destination Field for Standardized Data

If you delete any source fields from a standardization structure, you might also need to delete the corresponding destination fields.

To delete a destination field for standardized data

1 In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **free-form-texts-to-standardize** element in the **StandardizationConfig** element.

3 In the **group** element from which destination fields need to be removed, delete all text between and including the start and end **target-mapping** tags that define the field.

Using the example below, to delete the URL field, delete the boldface text.

```
<group standardization-type="BusinessName" domain-selector=
        "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
    <unstandardized-source-fields>
        ...
    </unstandardized-source-fields>
    <standardization-targets>
        <target-mapping>
            <standardized-object-field-id>PrimaryName
            </standardized-object-field-id>
            <standardized-target-field-name>Company.PrimaryName_Name
            </standardized-target-field-name>
        </target-mapping>
        <target-mapping>
            <standardized-object-field-id>UrlTypeKeyword
            </standardized-object-field-id>
            <standardized-target-field-name>Company.PrimaryName_Url
```

```
                </standardized-target-field-name>
            </target-mapping>
        </unstandardized-source-fields>
    </group>
```

Note: *Each standardization structure must have at least one destination field defined for standardized data. If a structure does not contain any fields that need to be standardized, you can delete the entire structure, as described previously in **"Deleting a Standardization Structure"**.*

4 Save and close the file.

6.6.3 Defining Phonetic Encoding

To specify that certain data be converted into its phonetic version before being processed by the match engine, you need to specify the source and destination fields for the data and the type of phonetic encoding to use, such as NYSIIS or Soundex. eView uses the phonetic encoders defined in the **PhoneticEncodersConfig** element to determine how the data is phonetically encoded. You can customize phonetic encoding by performing the following actions:

- **Defining Fields for Phonetic Encoding** on page 95
- **Deleting Fields Defined for Phonetic Conversion** on page 96

Defining Fields for Phonetic Encoding

Some of the data in an object must be converted into its phonetic encoding before the match process occurs. You must specify the fields you want to be phonetically encoded and the type of phonetic encoder, such as NYSIIS or Soundex, to use for each field.

To define fields for phonetic encoding

1 In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **phoneticize-fields** element in the **PhoneticEncodersConfig** element.

3 Do any of the following:

- To modify information about an existing field's phonetic encoding, change the value of any of the elements described in Table 16.
- To add a new field to be phonetically encoded, create a new **phoneticize-field** element within the **phoneticize-field**s element, and then define the elements described in Table 16 in the new tags. For example:

```
<phoneticize-fields>
    <phoneticize-field>
        ...
    </phoneticize-field>
    <phoneticize-field>
        <unphoneticized-source-field-name>Person.FirstName
        </unphoneticized-source-field-name>
        <phoneticized-target-field-name>Person.FirstNamePhoneticCode
        </phoneticized-target-field-name>
        <encoding-type>Soundex</encoding-type>
    </phoneticize-field>
</phoneticize-fields>
```

4 Save and close the file.

Table 16 phoneticize-field Elements

Element	Description
unphoneticized-source-field-name	The ePath of the source field in the system object from which the value to phoneticize will be retrieved (for example, **Person.Address[*].StreetName**). ***Note:*** *This can refer to the original field or to a standardized or normalized field.*
phoneticized-target-field-name	The ePath of the field in which the phoneticized value will be saved in the system object.
phoneticized-object-field-id	A field ID to identify the field to the phonetic encoder. This is only used in implementations with Ascential's INTEGRITY match engine.
encoding-type	The phonetic encoding to use for this field, such as NYSIIS or Soundex. This must correspond to an **encoding-type** configured for the desired encoder in the **PhoneticEncodersConfig** element.

Deleting Fields Defined for Phonetic Conversion

If a field currently defined for phonetic conversion does not require phonetic encoding, you can delete the field from the phonetic conversion structure.

To delete fields defined for phonetic conversion

1 In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **phoneticize-fields** element in the **PhoneticEncodersConfig** element.

3 Do either of the following:

- To delete a field currently specified for phonetic conversion, delete all text between and including the start and end **phoneticize-field** tags that define the field.

  Using the example below, to delete the first name phonetic field, delete the boldface text.

```
<phoneticize-fields>
    <phoneticize-field>
        <unphoneticized-source-field-name>Person.LastName
        </unphoneticized-source-field-name>
        <phoneticized-target-field-name>Person.LastNamePhoneticCode
        </phoneticized-target-field-name>
     <encoding-type>NYSIIS</encoding-type>
    </phoneticize-field>
    <phoneticize-field>
        <unphoneticized-source-field-name>Person.FirstName
        </unphoneticized-source-field-name>
        <phoneticized-target-field-name>Person.FirstNamePhoneticCode
        </phoneticized-target-field-name>
        <encoding-type>Soundex</encoding-type>
    </phoneticize-field>
</phoneticize-fields>
```

- To delete all fields currently specified for phonetic conversion, delete all text between, but not including, the **phoneticize-fields** tags.

4 Save and close the file.

6.7 Matching Configuration

The **MatchingConfig** section of the Match Field file defines the match string passed to the match engine for probabilistic weighting. By default, the fields defined for matching are the fields you specified for matching in the eView Wizard. You can modify the match string if necessary.

If you do modify the match string, you may need to make corresponding changes to the match engine configuration files as well. For more information about modifying these files, see the appropriate match engine implementation guide.

To customize the matching configuration, perform any of the following actions:

- **Defining the Match Object** on page 97
- **Configuring the Match String** on page 99

Important: *SeeBeyond recommends that no changes be made to the matching configuration once the master index is in production.*

6.7.1 Defining the Match Object

For the master index to determine matching weights between existing and incoming records, it must know which objects in each record contain the fields to use for matching. In the **MatchingConfig** section, you can create a new object, modify an existing object, and delete a match object. The type of match object you specify must correspond to an object defined in the Object Definition file.

- **Creating a Match Object** on page 98
- **Modifying a Match Object** on page 98
- **Deleting a Match Object** on page 98

Creating a Match Object

A default match object is predefined based on the match type information you specified in the eView Wizard. If no match types were defined using the wizard, or if you need to delete the existing match object, you can create a new match object.

To create a match object

1. In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **MatchingConfig** element.
3. Create a new **match-system-object** element in the **MatchingConfig** element, and then create and name a new **object-name** element. For example:

```
<MatchingConfig module-name="Matching" parser-class=
"com.stc.eindex.configurator.impl.matching.MatchingConfiguration">
    <match-system-object>
        <object-name>Company</object-name>
    </match-system-object>
</MatchingConfig>
```

Note: *The object name specified must be defined in the Object Definition file.*

4. Define the fields that will be used for matching in the match object, as described in **"Creating a Match String"**.
5. Save and close the file.

Modifying a Match Object

If you want to use an existing match object structure to define a match string for a different object, you can modify the match object name and associated fields.

To modify a match object

1. In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **MatchingConfig** element, and then scroll to the appropriate **match-system-object** element.
3. To change the object on which matching is performed, modify the value of the **object-name** element.
4. To specify the fields to use for matching, perform any of the procedures under **"Configuring the Match String"**.
5. Save and close the file.

Deleting a Match Object

If a match object has been defined in error, you can delete that match object and any associated fields from the matching structure. The matching structure must contain at least one match object.

To delete a match object

1 In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **MatchingConfig** element.

3 Delete all text between and including the **match-system-object** element that contains the object to be deleted.

   Using the following example, to delete the Person match object, delete all text below.

```
<match-system-object>
    <object-name>Person</object-name>
    <match-columns>
        <match-column>
         <column-name>Enterprise.SystemSBR.Person.FirstName
         </column-name>
         <match-type>FirstName</match-type>
        <match-column>
    </match-columns>
</match-system-object>
```

4 Save and close the file.

6.7.2 Configuring the Match String

Once you define the objects that contain the fields to be included in the match string, you can create and configure the fields for each object. The fields you specify must correspond to fields defined in the Object Definition file, and can include any fields in the object structure.

If you add or modify fields in the match string, be sure to modify the match engine configuration files accordingly. For more information, see the appropriate match engine implementation guide. Perform any of the following actions to configure the match string.

- **Creating a Match String** on page 99
- **Adding a Field to a Match String** on page 100
- **Modifying a Match String Field** on page 101
- **Deleting a Field from a Match String** on page 101

Creating a Match String

To create a match string, you need to define and configure the fields to be used in the matching process.

To create a match string

1 In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

2 In the **MatchingConfig** element, scroll to the **match-system-object** in which you want to create the match string.

3. Create a **match-columns** element after the **object-name** element.
4. In the new **match-columns** element, create a new **match-column** element.
5. In the new **match-column** element, create and define the elements described in Table 17. For example:

```
<match-system-object>
    <object-name>Address</object-name>
    <match-columns>
        <match-column>
         <column-name>Enterprise.SystemSBR.Person.Address.StreetName
         </column-name>
         <match-type>StreetName</match-type>
        </match-column>
    </match-columns>
</match-system-object>
```

Note: *For the SeeBeyond Match Engine, you must specify each parsed field to use for matching to match on standardized fields. For INTEGRIY, you must specify the unparsed source field to match on standardized fields.*

6. Repeat steps 4 and 5 for each field to add to the match string.
7. Save and close the file.

Table 17 Match Column Elements

Element	Description
column-name	The fully qualified field name that defines the location of each field on which to match (for example, **Enterprise.SystemSBR.Person.Address.City**).
match-type	An ID that is specific to the match engine and identifies the field to the match engine. This value must correspond to a match type defined for the match engine in use. For more information, see the appropriate match engine implementation guide.

Adding a Field to a Match String

Once you create a match string, you can add new fields to the match string if needed. This should only be done prior to moving to production. Otherwise, unexpected matching results may occur

To add a field to a match string

1. In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.
2. In the **MatchingConfig** section, scroll to the **match-system-object** element to which you want to add a new match field.
3. In the **match-columns** element of the **match-system-object** element, create a new **match-column** element.

```
<match-system-object>
    <object-name>Address</object-name>
    <match-columns>
        <match-column>
         <column-name>Enterprise.SystemSBR.Person.Address.StreetName
         </column-name>
         <match-type>StreetName</match-type>
        </match-column>
        <match-column>
        </match-column>
    </match-columns>
</match-system-object>
```

4 In the new **match-column** element, create and define the elements described in **Table 17 on page 100**. For example:

```
<match-system-object>
    <object-name>Address</object-name>
    <match-columns>
        <match-column>
         <column-name>Enterprise.SystemSBR.Person.Address.StreetName
         </column-name>
         <match-type>StreetName</match-type>
        </match-column>
        <match-column>
         <column-name>Enterprise.SystemSBR.Person.Address.HouseNo
         </column-name>
         <match-type>HouseNumber</match-type>
        </match-column>
    </match-columns>
</match-system-object>
```

5 Repeat steps 3 and 4 for each field you need to add.

6 Save and close the file.

Modifying a Match String Field

Once you define a match string, you can modify information about the fields in the match string as necessary. This should only be done prior to moving to production. Otherwise, unexpected matching results may occur.

To modify a match string field

1 In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

2 In the **MatchingConfig** section, scroll to the **match-system-object** to modify.

3 In the **match-column** element for the field you want to modify, change the value of either of the elements listed in **Table 17 on page 100**.

4 Save and close the file.

Deleting a Field from a Match String

If a match string contains a field that should not be used for probabilistic matching, you can delete that field from the match string.

To delete a field from a match string

1. In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.
2. In the **MatchingConfig** section, scroll to the **match-system-object** from which you want to delete a field.
3. Delete all text between and including the **match-column** element defining the field you want to delete.

   Using the example below, to delete the ZipCode field, delete the boldface text.

```
<match-system-object>
    <object-name>Address</object-name>
    <match-columns>
        <match-column>
         <column-name>Enterprise.SystemSBR.Person.Address.StreetName
         </column-name>
         <match-type>StreetName</match-type>
        </match-column>
        <match-column>
         <column-name>Enterprise.SystemSBR.Person.Address.HouseNo
         </column-name>
         <match-type>HouseNumber</match-type>
        </match-column>
```

4. Save and close the file.

6.8 MEFA Configuration

The **MEFAConfig** section of the Match Field file defines the components used during the standardization and matching processes. The following MEFA sections can be configured to use different types of the following components:

- Match Engine
- Standardization Engine
- Phonetic Encoders
- Matching BlockPicker
- Matching PassController

The match and standardization engines must both be from the same provider, (for example, either from SeeBeyond or Ascential, but not a combination of both).

You can perform the following actions to configure the MEFA.

- **Specifying a Block Picker Class** on page 103
- **Specifying a Pass Controller Class** on page 103
- **Configuring the Standardization Engine** on page 104
- **Configuring the Match Engine** on page 104

Specifying a Block Picker Class

The block picker determines the blocking strategy to use for each match pass. Blocking strategies define how to create the queries that check the database for a subset of the records to be used for matching. The default Block Picker has access to the match results from previous match passes, as well as lists of applicable blocking definitions that have been executed and of those that have not.

The default Block Picker class is **com.stc.eindex.matching.impl.PickAllBlocksAtOnce**, which selects all blocks during the first pass.

To specify the Block Picker class

1. In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **block-picker** element in the **MatchingConfig** section.
3. In the **class-name** element, specify the Java class for the block picker you want to use, using the fully qualified class name.

```
<block-picker>
    <class-name>com.stc.eindex.matching.impl.PickAllBlocksAtOnce
    </class-name>
</block-picker>
```

4. Save and close the file.

Specifying a Pass Controller Class

The matching process can be executed in multiple stages. Each query block in the blocking query is executed in a separate match pass. After a block is evaluated, the pass controller determines if the results found are sufficient or if the query should continue by performing another match pass.

The default pass controller is **com.stc.eindex.matching.impl.PassAllBlocks**. This class instructs the match engine to continue calculating match weights until all applicable block definitions have been processed.

To specify the Pass Controller class

1. In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **pass-controller** element in the **MatchingConfig** section.
3. In the **class-name** element, specify the Java class for the Pass Controller you want to use, using the fully qualified class name.

```
<pass-controller>
    <class-name>com.stc.eindex.matching.impl.PassAllBlocks
    </class-name>
</pass-controller>
```

4. Save and close the file.

Configuring the Standardization Engine

eView supports standardization engines from different vendors depending on the adapter configured to communicate with the engine. SeeBeyond provides classes for using the SeeBeyond Match Engine or Ascential INTEGRITY. You can implement a custom standardization engine as well. The standardization engine configuration is defined by **standardizer-api** and **standardizer-config** elements.

To configure the standardization engine

1. In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **standardizer-api** element in the **MatchingConfig** section.
3. Specify the Java class for the standardization adapter to use, using the fully qualified class name.

   To implement one of the standardization engines provided with eView, enter one of the classes described in **Table 18 on page 104**.

```
<standardizer-api>
     <class-name>com.stc.eindex.matching.adapter.SbmeStandardizerAdapter
     </class-name>
</standardizer-api>
```

4. In the **standardizer-config** element, specify the Java class for the configuration of the standardization adapter, using the fully qualified class name as shown below.

```
<standardizer-config>
     <class-name>
     com.stc.eindex.matching.adapter.SbmeStandardizerAdapterConfig
     </class-name>
</standardizer-config>
```

   To implement the SeeBeyond Match Engine for standardization, enter **com.stc.eindex.matching.adapter.SbmeStandardizerAdapterConfig**.

   To implement INTEGRITY for standardization, enter **com.stc.eindex.matching.adapter.IntegrityStandardizerAdapterConfig**.

5. Save and close the file.

Table 18 Standardization Engine API Classes

class-name	Description
com.stc.eindex.matching.adapter.SbmeStandardizerAdapter	Specifies the SeeBeyond standardization engine.
com.stc.eindex.matching.adapter.IntegrityStandardizerAdapter	Specifies INTEGRITY as the standardization engine.

Configuring the Match Engine

eView supports different match engines depending on the adapter configured to communicate with the engine. SeeBeyond provides classes for using the SeeBeyond Match Engine or Ascential INTEGRITY. You can implement a custom match engine as well. The match engine configuration is defined by the **matcher-api** and **matcher-config** elements.

To configure the match engine

1 In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **matcher-api** element in the **MatchingConfig** section.

3 Specify the Java class for the matching adapter to use, using the fully qualified class name.

   To implement one of the match engines provided with eView, enter one of the classes described in Table 19.

```
<matcher-api>
    <class-name>com.stc.eindex.matching.adapter.SbmeMatcherAdapter
    </class-name>
</matcher-api>
```

4 In the **matcher-config** element, specify the Java class for the configuration of the matching adapter, using the fully qualified class name.

   To implement one of the match engines provided with eView, enter one of the classes described in Table 20.

```
<matcher-config>
    <class-name>com.stc.eindex.matching.adapter.SbmeMatcherAdapterConfig
    </class-name>
</matcher-config>
```

5 Save and close the file.

Table 19 Match Engine API Classes

class-name	Description
com.stc.eindex.matching.adapter.SbmeMatcherAdapter	Specifies the SeeBeyond Match Engine as the match engine.
com.stc.eindex.matching.adapter.IntegrityMatcherAdapter	Specifies INTEGRITY as the match engine.

Table 20 Match Engine Configuration Classes

class-name	Description
com.stc.eindex.matching.adapter.SbmeMatcherAdapterConfig	Provides the SeeBeyond Match Engine with the relevant configuration information.
com.stc.eindex.matching.adapter.IntegrityMatcherAdapterConfig	Provides INTEGRITY with the relevant configuration information.

6.9 Phonetic Encoding

eView provides configurable phonetic encoding capabilities. Phonetic encoding is used to retrieve records with similar field values from the database for matching. If you implement the SeeBeyond Match Engine, you can modify the phonetic encoders in the **PhoneticEncodersConfig** section of the Match Field file.

Defining Phonetic Encoders

Each type of phonetic encoder provided with eView is listed as an **encoder** element in the **PhoneticEncodersConfig** section of the Match Field file. In the **StandardizationConfig** section, you can specify which of these encoders to use for each phonetic field.

To define phonetic encoders

1. In the Project Explorer pane of the Enterprise Designer, double-click **Match Field** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **PhoneticEncodersConfig** section.
3. To define a new phonetic encoder, create a new **encoder** element, and then define the elements described in Table 21.

   The default encoder definitions are shown below.

```
<encoder>
     <encoding-type>NYSIIS</encoding-type>
     <encoder-implementation-class>com.stc.eindex.phonetic.impl.Nysiis
     </encoder-implementation-class>
</encoder>
<encoder>
     <encoding-type>Soundex</encoding-type>
     <encoder-implementation-class>com.stc.eindex.phonetic.impl.Soundex
     </encoder-implementation-class>
</encoder>
```

4. Save and close the file.

Table 21 PhoneticEncodersConfig Elements

Element	Description
encoding-type	The name of the phonetic encoder, such as NYSIIS or Soundex.
encoder-implementation-class	The fully qualified name of the Java class that determines the behavior of the phonetic encoder. Table 22 lists and describes the default classes.

Table 22 Default Phonetic Encoder Classes

class-name	Description
com.stc.eindex.phonetic.impl.Nysiis	NYSIIS encoding
com.stc.eindex.phonetic.impl.Soundex	Soundex encoding

Chapter 7

Best Record Configuration

The Best Record file configures the Update Manager. The Update Manager defines logic used when updates are made to the master index database and when data is populated into the SBR. This chapter describes the Update Manager, the Best Record file, and provides instructions for customizing the Best Record file for your processing requirements.

7.1 About the Update Manager

The Update Manager contains the logic used to generate the single best record (SBR) for a given object. The SBR is defined by a mapping of fields from external systems to the SBR, allowing you to define the fields from each system that is kept in the SBR. For each field in the SBR, an ePath denotes the location in the external system records from which the value is retrieved. Since there can 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 (by default, the record's update date and time is always taken into account).

The Update Manager also specifies any custom Java classes to be used for different types of update transactions, such as merges, unmerges, changes to existing records, and new record inserts.

7.2 The Survivor Calculator and the SBR

The survivor calculator generates and updates the SBR for each record. The SBR for an enterprise object is created from what is considered to be the most reliable information contained in each system record for a particular object. The information used from each local system to populate the SBR is determined by the survivor calculator defined in the Update Manager. The fields defined in the survivor calculator also define the fields contained in the SBR. You can configure the survivor calculator to determine the best fields for the SBR from a combination of all the source system records. The survivor calculator can consider factors such as the relative reliability of a system, how recent the data is, and whether data entered from the EDM overwrites data entered from any other system.

The survivor calculator consists of the rules defined for the survivor helper and the weighted calculator.

Important: *Phonetic and standardized fields do not need to be defined in the Best Record file since their field values are determined by the standardization engine for the SBR.*

7.3 Update Manger Components

The logic that determines how the fields in the SBR are populated and how certain updates are performed is highly configurable in eView, allowing you to design and develop the match strategy that best suits your processing requirements. Configuring the best record consists of customizing these components.

- Survivor Helper
- Weighted Calculator
- Update Manager Policies

7.3.1 Survivor Helper

The survivor helper defines a list of fields on which survivor calculation is performed. Each field is called a “candidate field”. For each candidate field, you specify whether to use the default survivor calculation strategy or a custom strategy. The survivor helper must list each field contained in the SBR; any SBR fields that are not listed here will not be populated in the SBR.

For each field, you can specify system fields to be taken into consideration as well as a specific survivorship strategy. There are three basic strategies provided by eView to determine survivorship for each field.

- Default Strategy
- Weighted Strategy
- Union Strategy

You can also define and implement custom strategies.

Default Strategy

This strategy maps fields directly from the local system records to the SBR. When you specify the default survivor strategy for a field, you must also specify the parameter that defines the source system. For example, if you specify the default survivor calculator for the field “Person.LastName” and define the preferred system as “SystemA”, the last name field in the SBR is always taken from SystemA (unless the value is overridden in the EDM).

The default survivor strategy is **com.stc.eindex.survivor.impl.DefaultStrategy**.

Weighted Strategy

This strategy is the most complex survivor strategy, and uses a combination of weighted calculations to determine the most reliable source of data for each field. This strategy is highly customizable, and you can define which calculation or set of calculations to use for each field. The calculations can be based on the update date of the data, system reliability, and agreement between systems. In the default configuration of the file, the calculations are defined in the WeightedCalculator section of the file.

The weighted survivor strategy is **com.stc.eindex.survivor.impl.WeightedSurvivorStrategy**. You can define general weighted calculations to be performed by default for each field, and you can define specialized calculations to be performed for specific fields.

Union Strategy

This strategy combines the data from all source systems to populate the fields in the SBR for which this strategy is specified. For example, if you store aliases for person names in the database, you want to store all possible alias records and not just the “best” alias information. In order to do this, specify the union strategy for the alias object. This means that all alias information from all source systems is stored in the SBR.

The union strategy is applied to entire objects, rather than to fields. The union strategy combines all child objects from an enterprise objects source systems to populate the SBR. If the source systems contain two or more instances of a specific child object (for example, two home telephone numbers), the union strategy only populates the most current child object in the SBR. For example, if the union strategy is assigned to the address object and each address object is identified by a unique key (such as the address type), the SBR only contains the most current address record of each address type (for example, one home address, one office address, and so on).

The union strategy is **com.stc.eindex.survivor.impl.UnionSurvivorStrategy**.

7.3.2 Weighted Calculator

By default, the weighted calculator implements the weighted strategy defined above. strategy you can configure for each field. Use the WeightedCalculator section to define conditions and weights that determine the best information with which to populate the SBR. The weighted calculator selects a single value for the SBR from a set of system fields. The selection process is based on the different qualities defined for each field.

The weighted calculator defines two sets of rules. The *default rules* apply to all fields in a record except those fields for which rules are specifically defined. The *candidate rules* only apply to those fields for which they are specifically defined. If you modify the default rules, the changes will apply to all fields except the fields for which candidate rules are defined.

Weighted Calculator Strategies

You can define several strategies to help the weighted calculator determine the best information to populate into each field of the SBR. Each of these strategies is defined by a quality, a preference, and a utility. The quality defines the type of weighted calculation to perform, the preference indicates the source being rated, and the utility indicates the reliability. You can define multiple strategies for each field, and a linear summation on the utility score of each strategy determines the best value to populate in the SBR field.

The weighted calculator strategies include:

- SourceSystem
- SystemAgreement
- MostRecentModified

SourceSystem

This strategy indicates the best source system for a field, and is used when the quality of the field in question depends on its origin. For example, to indicate that the data from SystemA for a specific field is of a higher quality than SystemB, define a SourceSystem quality for “SystemA” and one for “SystemB”. Then assign SystemA a higher utility value (85.0, for example) and SystemB a lower utility value (30.0, for example). This indicates that SystemA is a more reliable source for the field. If both SystemA and SystemB contain the specified field, the value from SystemA is populated into the SBR. If the field is empty in SystemA but the field in SystemB contains a value, then the value from SystemB is used.

SystemAgreement

This strategy pro-rates the utility score based on the number of systems whose values for the specified field are in agreement. For example, if the first name field for SystemA is “John”, for SystemB is “John”, and for SystemC is “Jon”, SystemA and SystemB together receive two-thirds of the utility score, while SystemC only receives one-third. The value populated into the SBR is “John”. You do not need to define a preference for the **MostRecentModified** strategy, but you must define source systems.

MostRecentModified

This strategy ranks the field values from the source systems in descending order according to the time that the object was last modified. The value populated in the SBR comes from the most recently modified object. You do not need to define a preference for the **MostRecentModified** strategy, but you must define a utility.

7.3.3 Update Manager Policies

The update manager policies specify certain Java classes to use in each type of update transaction to specify additional processing that is not included in the standard eView instance. You can define custom update policies using the Custom Plug-ins function in the eView Project in Enterprise Designer, which appears after the Project is generated. Once all custom plug-ins are defined, eView builds and compiles the custom Java code. The Java classes defining the update policies are specified for eView in the **UpdateManagerConfig** element of the Best Record file.

Update Policies

There are seven types of update policies defined in the Update Manager.

- **Enterprise Merge Policy**
  The enterprise merge policy defines additional processing to perform when two enterprise objects are merged. This policy is defined by the **EnterpriseMergePolicy** element.
- **Enterprise Unmerge Policy**
  The enterprise unmerge policy defines additional processing to perform when an unmerge transaction occurs. This policy is defined by the **EnterpriseUnmergePolicy** element.
- **Enterprise Update Policy**
  The enterprise update policy defines additional processing to perform when a record is updated. This policy is defined by the **EnterpriseUpdatePolicy** element.
- **Enterprise Create Policy**
  The enterprise create policy defines additional processing to perform when a new record is inserted into the master index database. This policy is defined by the **EnterpriseCreatePolicy** element.
- **System Merge Policy**
  The system merge policy defines additional processing to perform when two system objects are merged. This policy is defined by the **SystemMergePolicy** element.
- **System Unmerge Policy**
  The system unmerge policy defines additional processing to perform when system objects are unmerged. This policy is defined by the **SystemUnmergePolicy** element.
- **UndoAssumeMatchPolicy**
  The undo assume match policy defines additional processing to perform when an assumed match transaction is reversed. This policy is defined by the **UndoAssumeMatchPolicy** element.

Update Policy Flag

The update policy section includes a flag that can prevent the update policies from being carried out if no changes were made to the existing record. When set to “true”, the **<SkipUpdateIfNoChange>** flag prevents the update policies from being performed when no changes are made to an existing file. Setting the flag to true helps increase performance when processing a large number of updates.

7.4 The Best Record File

The properties for the update process are defined in the Best Record file in XML format. Some of the information entered into the default configuration file is based on the fields defined in the eView Wizard, and some is standard across all implementations. For most implementations, this file will require customization.

7.4.1 Modifying the Best Record File

You can customize the configuration of the Update Manager by modifying the Best Record file. You can modify the Best Record file at any time, but this is not recommended after moving into production. The configuration controls how the SBR for each object is created, and modifying the file can cause discrepancies in how SBRs are formed before and after the modifications. It may also cause discrepancies in match results, since matching is performed against the SBR. You must regenerate the Project and reactivate the deployment after modifying this file. You can either apply the changes immediately to the Logical Host, or restart the host to pick up the changes.

Before making any changes to this file, make sure you are familiar with the procedures described under **"Modifying Configuration Files" on page 24**. The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes.

7.4.2 Best Record File Structure

The Best Record file is written in XML, and is automatically generated by the eView Wizard. This file can be modified using the Enterprise Designer XML editor, which is a standard NetBeans editor. This section describes the structure of the XML file and general requirements and constraints. There are four primary components of the Best Record file. The first component, XML File Information, is standard across all implementations.

- XML File Information
- Survivor Helper
- Weighted Calculator
- Update Manager Policies

XML File Information

The first lines in the Best Record file represent information about the schema instance and namespace. You should not need to modify these components.

Survivor Helper

The **SurvivorHelperConfig** section of contains two primary elements: **default-survivor-strategy** and **candidate-definitions**. The first two elements of this section define the module name and the Java class that parses this section of the file; these two elements should not be changed.

default-survivor-strategy

You can define multiple survivor strategies, and use different strategy combinations for the candidate fields. Any field that is not assigned a specific survivor strategy in the **candidate-definitions** list uses the default survivor strategy. The **default-survivor-strategy** element defines the Java class to use as the default survivor strategy, and defines parameters for the that class.

The **default-survivor-strategy** element contains a **strategy-class** element and a **parameters** element. The **strategy-class** element names the Java class of the default strategy, and the **parameters** element contains a list of optional parameters for the specified class. The default parameter specifies the name of the section in the Best Record file that defines the default class, as shown in the sample below.

```
<default-survivor-strategy>
     <strategy-class>com.stc.eindex.survivor.impl.WeightedSurvivorStrategy
     </strategy-class>
     <parameters>
          <parameter>
               <parameter-name>ConfigurationModuleName</parameter-name>
               <parameter-type>java.lang.String</parameter-type>
               <parameter-value>WeightedSurvivorCalculator</parameter-value>
          </parameter>
     </parameters>
</default-survivor-strategy>
```

Table 23 on page 119 provides a list of available parameter elements and their values for both the **DefaultSurvivorStrategy** and the **WeightedSurvivorStrategy** classes. The weighted survivor strategy (described in **"Weighted Calculator" on page 114**) is automatically defined as the default. The Java class name for this strategy is **com.stc.eindex.survivor.impl.WeightedSurvivorStrategy**.

candidate-definitions

The **candidate-definitions** element contains a list of fields defined by **candidate-field** elements. The fields are named by a **name** attribute in the **candidate-field** elements. A candidate field may include a **description**, a **system-fields**, or a **survivor-strategy** element. These elements are all optional.

- **description**—This element is a short description of the candidate field.
- **system-fields**—This element contains one element, **field-name**, that specifies a field other than the candidate field that is used when determining the value for the SBR. One example of this would be using the last update date of the record. This element is not currently used by any of the survivor strategies provided with eView, but may be useful when defining custom strategies.
- **survivor-strategy**—This element specifies a survivor strategy to use for the given field in place of the **default-survivor-strategy**. If no **survivor-strategy** element is specified for a given field, the **default-survivor-strategy** is used for that field.

Below is a sample of the above elements. In this sample, the first three fields are updated using the strategy defined as the default strategy; the **Mstatus** field is updated using a custom strategy (which would need to be defined using the Custom Plug-ins function); and **Address** objects are updated using the union strategy.

```
<candidate-definitions>
     <candidate-field name="Person.LastName"/>
     <candidate-field name="Person.FirstName"/>
     <candidate-field name="Person.MiddleName"/>
     <candidate-field name="Person.Mstatus">
          <survivor-strategy>
           <strategy-class>com.stc.eindex.survivor.impl.MySurvivorStrategy
           </strategy-class>
          </survivor-strategy>
     </candidate-field>
     <candidate-field name="Person.Address[*].*">
          <survivor-strategy>
           <strategy-class>com.stc.eindex.survivor.impl.UnionSurvivorStrategy
           </strategy-class>
```

```
            </survivor-strategy>
        </candidate-field>
```

Weighted Calculator

The **WeightedCalculator** section of the Best Record file contains two primary elements: **candidate-field** and **default-parameters**. The **default-parameters** element defines the default weighted calculator logic for all fields except those whose logic is defined in the **candidate-field** element.

By default, this section is specified as the default strategy in the **default-survivor** strategy element (described earlier in this section).

default-parameters

The **default-parameters** element contains a list of **parameter** elements. Each **parameter** element contains a **quality**, **utility**, and, optionally, a **preference** element. These elements define various rules, or survivor strategies, for determining the values to be populated into the SBR. Table 24 on page 122 lists the names and descriptions of the parameters you can specify to define the weighted calculator.

Below is a sample of the weighted strategy parameters. Using this sample, if there is a value in either system record but not in the other system record, that value is used in the SBR regardless of update date. If there is a value in both system records, and they were updated at the same time, the SAP field value is used (80.0>30.0). If there is a value in both system records, but CDW was the most recently modified, the value from CDW is populated into the SBR ((30.0+70.0)>80.0)

```
<default-parameters>
    <parameter>
        <quality>SourceSystem</quality>
        <preference>SAP</preference>
        <utility>80.0</utility>
    </parameter>
    <parameter>
        <quality>MostRecentModified</quality>
        <utility>70.0</utility>
    </parameter>
    <parameter>
        <quality>SourceSystem</quality>
        <preference>CDW</preference>
        <utility>30.0</utility>
    </parameter>
</default-parameters>
```

candidate-field

The **candidate-field** element allows you to customize the logic of the weighted calculator for certain fields. The logic you specify here overrides the logic defined in the **default-parameters** section, but only for the fields specified. You need to define parameters for each field for which you want to override the default parameters. This section consists of a list of fields (**candidate-field** elements), each identified by a **name** attribute and each defining the survivor strategies for one field. The name must match the name of the field for which you want to define override logic. Each **candidate-field** element contains the same parameter elements as described for **default-parameters** above. These elements are described in Table 24 on page 122. A sample is shown below.

```
<candidate-field name="Person.SSN">
```

```
        <parameter>
            <quality>SourceSystem</quality>
            <preference>CPDA</preference>
            <utility>70.0</utility>
        </parameter>
        <parameter>
            <quality>MostRecentModified</quality>
            <utility>75.0</utility>
        </parameter>
        <parameter>
            <quality>SourceSystem</quality>
            <preference>ACCT</preference>
            <utility>50.0</utility>
        </parameter>
    </candidate-field>
```

Update Manager Policies

The **UpdateManagerConfig** section of the Best Record file allows you to define a list of Java classes to manage custom processing for different types of updates. You can create the custom classes to implement in the **Custom Plug-ins** function of the eView Project in the Enterprise Designer, and then specify those classes here.

Following is a list of **UpdateManagerConfig** elements. These elements are described in **"Update Manager Policies" on page 110**.

- EnterpriseMergePolicy
- EnterpriseUnmergePolicy
- EnterpriseUpdatePolicy
- EnterpriseCreatePolicy
- SystemMergePolicy
- SystemUnmergePolicy
- UndoAssumeMatchPolicy
- SkipUpdateIfNoChange

Note: *The first two elements of the **UpdateManagerConfig** section define the module name and the Java class that parses this section of the file. These elements should not be changed.*

Below is a sample of an implementation of update manager policies along with the update policy flag.

```
<EnterpriseMergePolicy>com.stc.eindex.user.CustomMergePolicy
</EnterpriseMergePolicy>
<EnterpriseUnmergePolicy>com.stc.eindex.user.CustomUnmergePolicy
</EnterpriseUnmergePolicy>
<EnterpriseUpdatePolicy>com.stc.eindex.user.CustomUpdatePolicy
</EnterpriseUpdatePolicy>
<EnterpriseCreatePolicy>com.stc.eindex.user.CustomCreatePolicy
</EnterpriseCreatePolicy>
<SystemMergePolicy>com.stc.eindex.user.CustomSystemMergePolicy
</SystemMergePolicy>
<SystemUnmergePolicy>com.stc.eindex.user.CustomSystemUnmergePolicy
</SystemUnmergePolicy>
<UndoAssumeMatchPolicy>com.stc.eindex.user.CustomUndoAsmMatchPolicy
```

```
        </UndoAssumeMatchPolicy>
        <SkipUpdateIfNoChange>true</SkipUpdateIfNoChange>
```

7.5 Customizing the Best Record File

There are three components of the Best Record file that must be customized for a master index implementation. Before you begin, make sure you know which fields to include in the SBR (typically, all but phonetic and standardized fields are configured here), and how those fields should be updated. Configuring this file consists of the following primary tasks.

- **Configuring the Survivor Helper** on page 116
- **Configuring the Weighted Calculator** on page 121
- **Configuring Update Policies** on page 125

All of these actions require that you use the Enterprise Designer XML editor. For more information about this editor, see **"Using the eView Editors" on page 21**. Make sure you are familiar with the procedures described under **"Modifying Configuration Files" on page 24** for checking the file in and out, saving changes, and validating the file.

7.5.1 Configuring the Survivor Helper

The **SurvivorHelper** element of the Best Record file defines how field values are populated into the single best record, along with the type of survivor strategy to use for each field. If no strategy is defined for a field, the default survivor strategy is used to populate that field in the SBR. Perform the following tasks to configure the Survivor Helper.

- **Specifying the Survivor Helper** on page 116
- **Specifying a Default Survivor Strategy** on page 117
- **Configuring the Default Survivor Strategy** on page 118
- **Specifying Candidate Fields** on page 119
- **Deleting Candidate Fields** on page 120
- **Defining a Survivor Strategy for a Field** on page 120

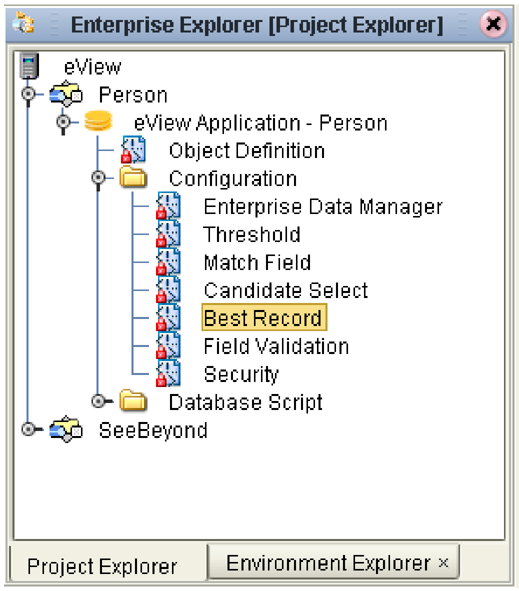
Specifying the Survivor Helper

The survivor helper class determines how to retrieve values from system records and how to set them in the SBR. The default class is **com.stc.eindex.survivor.impl.DefaultSurvivorHelper**, which uses the ePath method described in Chapter 2 to retrieve and set the values. You can create a custom survivor helper class to support other methods for retrieving and setting values. If you implement a custom survivor helper class, it must extend **com.stc.eindex.survivor.AbstractSurvivorHelper**.

To specify the survivor helper

1. In the Project Explorer pane of the Enterprise Designer, double-click **Best Record** under the **Configuration** folder in the Project you want to modify.

Figure 6 Best Record Node



2. Scroll to the **helper-class** element in the **SurvivorHelperConfig** element.
3. Change the value of the **helper-class** element to the fully qualified name of new helper class. For example:

```
<helper-class>com.stc.eindex.survivor.impl.MySurvivorHelper</helper-
class>
```

4. Save and close the file.

Specifying a Default Survivor Strategy

The default survivor strategy specifies the name of the Java class that defines the survivor calculation strategy to use for most of the fields in the SBR. By defining a default strategy, you do not need to define a strategy for every candidate field; you only need to define a strategy for fields that do not use the default strategy.

Note: *If you create a customized class for the default survivor strategy, make sure the class implements* ***com.stc.eindex.survivor.SurvivorStrategyInterface*** *and is accessible by the EJB class loader.*

To specify a default survivor strategy

1. In the Project Explorer pane of the Enterprise Designer, double-click **Best Record** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **default-survivor-strategy** element in the **SurvivorHelperConfig** element.

3 To change the name of the default class, modify the value of the **strategy-class** element to the fully qualified name of the new default strategy class. For example:

```
<default-survivor-strategy>
    <strategy-class>com.stc.eindex.survivor.impl.MySurvivorStrategy
    </strategy-class>
</default-survivor-strategy>
```

4 Configure the strategy parameters, as described in **"Configuring the Default Survivor Strategy"** below.

5 Save and close the file.

Configuring the Default Survivor Strategy

Once you define a default survivor strategy, you may need to specify certain parameters for the strategy. One parameter is required for the **WeightedSurvivorStrategy** and for the **DefaultSurvivorStrategy**. If you create a custom strategy, additional parameters may be used.

Note: *If you create a custom strategy, it must implement **com.stc.eindex.survivor.SurvivorStrategyInterface** to be recognized by eView.*

To configure the default survivor strategy

1 In the Project Explorer pane of the Enterprise Designer, double-click **Best Record** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **default-survivor-strategy** element, and then to the **strategy-class** element.

3 Do any of the following:

- To modify an existing parameter value, scroll to the parameter you want to modify, and then change the value of the **parameter-value** element. For example:

```
<default-survivor-strategy>
    <strategy-class>com.stc.eindex.survivor.impl.MySurvivorStrategy
    </strategy-class>
    <parameters>
        <parameter>
            <parameter-name>ConfigModuleName</parameter-name>
            <parameter-type>java.lang.String</parameter-type>
            <parameter-value>MySurvivorCalculator</parameter-value>
        </parameter>
    </parameters>
</default-survivor-strategy>
```

- To add a new parameter, create a new **parameter** element within the **parameters** element, and then define the parameter elements described in **Table 23 on page 119**.
- To delete a parameter, scroll to the **parameters** element, and then delete all text between and including the **parameter** tags that define the parameter.
- To delete all parameters, delete the all text between and including the **parameters** element.

4 Save and close the file.

Table 23 Default Survivor Strategy Parameter Elements

Element	Value
description	This is an optional element that briefly describes the parameter.
parameter-name	The name of the parameter. ▪ For the **DefaultSurvivorStrategy**, this value is "preferredSystem". ▪ For the **WeightedSurvivorStrategy**, this value is "ConfigurationModuleName".
parameter-type	The Java data type for the parameter value. For both the **DefaultSurvivorStrategy** and the **WeightedSurvivorStrategy**, this value is "java.lang.String".
parameter-value	The value of the named parameter. ▪ For the **DefaultSurvivorStrategy**, this is the processing code of the source system from which the SBR field value is retrieved. ▪ For the **WeightedSurvivorStrategy**, this is the name of the **module-name** element that defines the weighted calculator to use as the default strategy (by default, **WeightedSurvivorCalculator**).

Specifying Candidate Fields

In order for a field to be populated in the SBR, that field must be defined in the candidate field list of the survivor helper. By default, all the fields that were specified in the eView Wizard file are also defined here. Any candidate fields defined for the SBR must also be defined in Object Definition. If you add a field to the Object Definition, you should also add the field here.

To specify a candidate field

1 In the Project Explorer pane of the Enterprise Designer, double-click **Best Record** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **candidate-definitions** element in the **SurvivorHelperConfig** element.

3 Do any of the following:

- To add a new candidate field, add a new **candidate-field** element within the **candidate-definitions** tags, and then name the new element using the ePath of the field. For example:

```
<candidate-definitions>
    <candidate-field name="Company.Name"/>
    <candidate-field name="Company.AccountNumber"/>
<candidate-definitions>
```

- To modify an existing candidate field, scroll to the **candidate-field** element you want to modify, and then change the name of the element.

4 If any of the updated fields do not use the default strategy, define a strategy for those fields, as described in **"Defining a Survivor Strategy for a Field"**.

5 Save and close the file.

Deleting Candidate Fields

Once a field is defined in the candidate field list, you can delete the field if you do not want to include the field in the SBR. If you delete a field from the Object Definition, make sure to delete the field here as well.

To delete a candidate field

1 In the Project Explorer pane of the Enterprise Designer, double-click **Best Record** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **candidate-definitions** element in the **SurvivorHelperConfig** element.

3 Delete all text between and including the appropriate **candidate-field** tags.

Using the example below, to delete the Religion field, delete the boldface text; to delete the Alias object, delete the plain text.

```
<candidate-field name="Person.Religion"/>
<candidate-field name="Person.Alias[*].*"/>
    <system-fields>
        <field-name>LastModified</field-name>
    </system-fields>
    <survivor-strategy>
        <strategy-class>com.stc.eindex.user.MyStrategy
        </strategy-class>
    </survivor-strategy>
</candidate-field>
```

Note: *You cannot delete all candidate fields; at a minimum, the match fields must be defined.*

4 Save and close the file.

Defining a Survivor Strategy for a Field

To use a strategy for a specific field other than the strategy defined in the **default-survivor-strategy** element, you must specify the new strategy for the appropriate **candidate-field** element. You do not need to specify a strategy for any fields using the default survivor strategy.

To define a survivor strategy for a field

1 In the Project Explorer pane of the Enterprise Designer, double-click **Best Record** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **candidate-definitions** element in the **SurvivorHelperConfig** element.

3 In the **candidate-field** element for which you want to specify a new strategy, create a new **survivor-strategy** element. For example:

```
<candidate-field name="Person.Alias[*].*">
    <survivor-strategy>
    </survivor-strategy>
```

```
</candidate-field>
```

4 In the new **survivor-strategy** element, create and name a new **strategy-class** element.

   Make sure to specify the fully qualified name of the Java class for the strategy. For example:

```
<candidate-field name="Person.Alias[*].*">
     <survivor-strategy>
          <strategy-class>com.stc.eindex.survivor.impl.UnionSurvivorStrategy
          </strategy-class>
     </survivor-strategy>
</candidate-field>
```

Note: *To specify the default survivor strategy for a field, make sure the corresponding **candidate-field** element does not contain a **survivor-strategy** element. If you implement a custom strategy class, that class must be defined using the Custom Plug-in function of the Project.*

5 Save and close the file.

7.5.2 Configuring the Weighted Calculator

The **WeightedCalculator** element defines the Java class used for weighted survivor calculations. If the weighted calculator is defined as the **default-survivor-strategy**, then the strategies you define here are used for all candidate fields for which no specific survivor strategy is defined. The weighted calculator defines a default strategy to use for most fields, and specialized strategies to use for specific fields.

Configuring the weighted calculator involves the following tasks.

- **Defining Custom Weighted Strategies** on page 121
- **Adding Default Weighted Calculator Parameters** on page 122
- **Modifying Weighted Calculator Parameters** on page 123
- **Deleting Weighted Calculator Parameters** on page 124

Note: *Before you begin the following procedures, make sure you have information about the data in your source systems, such as which systems contain the most accurate and current data. You should also analyze how relevant the update date of the object should be in determining the values for the SBR.*

Defining Custom Weighted Strategies

The **WeightedCalculator** element defines both default and custom survivor strategies. You can override the default weighted calculator strategy for certain fields by defining custom strategies for those fields in the **candidate-field** elements of the **WeightedCalculator** element.

To define custom weighted calculators

1 In the Project Explorer pane of the Enterprise Designer, double-click **Best Record** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **WeightedCalculator** element.

3 Add and name a new **candidate-field** element in the **WeightedCalculator**.

```
<WeightedCalculator module-name="WeightedSurvivorCalculator" parser-
class="com.stc.eindex.configurator.impl.WeightedCalculatorConfig">
    <candidate-field name="Person.DOB">
    </candidate-field>
```

4 Create one or more **parameter** elements for the new candidate field.

```
<candidate-field name="Person.DOB">
    <parameter>
    </parameter>
    <parameter>
    </parameter>
</candidate-field>
```

5 For each new **parameter** element, define the elements listed in Table 24.

```
<candidate-field name="Person.DOB">
    <parameter>
        <quality>SourceSystem</quality>
        <preference>CDI</preference>
        <utility>80.0</utility>
    </parameter>
    <parameter>
        <quality>MostRecentModified</quality>
        <utility>75.0</utility>
    </parameter>
</candidate-field>
```

6 Save and close the file.

Table 24 Weighted Calculator Parameter Elements

Element	Description
quality	The type of weighted calculation to perform, such as: ▪ SourceSystem ▪ SystemAgreement ▪ MostRecentModified For more information about these qualities, see **“Weighted Calculator Strategies” on page 110**.
preference	The preferred value for the specified quality. For example, if the quality is SourceSystem, the preference must be a source system code. This element is only needed for the SourceSystem quality.
utility	A value that indicates the reliability of the specified quality for determining the best field value for the SBR. You define the scale for the utility values.

Adding Default Weighted Calculator Parameters

The eView Wizard creates a default weighted strategy that defines a general weighting structure to be used by most fields. Unless custom weighted calculator strategies are defined for a field, the default strategies defined in the **default-parameters** element are used for each field using the weighted calculator.

To add default weighted calculator parameters

1. In the Project Explorer pane of the Enterprise Designer, double-click **Best Record** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **default-parameters** element in the **WeightedCalculator** element.
3. Create new a new **parameter** element within the **default-parameters** element, but outside any existing **parameter** elements.

```
<default-parameters>
    <parameter>
        <quality>SourceSystem</quality>
        <preference>CDA</preference>
        <utility>80.0</utility>
    </parameter>
    <parameter>
    </parameter>
</default-parameters>
```

4. In the new **parameter** element, define the elements listed in **Table 24 on page 122**. For example:

```
<default-parameters>
    <parameter>
        <quality>SourceSystem</quality>
        <preference>CDA</preference>
        <utility>80.0</utility>
    </parameter>
    <parameter>
        <quality>MostRecentModified</quality>
        <utility>75.0</utility>
    </parameter>
</default-parameters>
```

5. Save and close the file.

Modifying Weighted Calculator Parameters

Once a candidate field is specified and custom weighted calculators are defined for the field, you can modify the parameters. You can also modify any existing default weighted calculator parameters.

To modify weighted calculator parameters

1. In the Project Explorer pane of the Enterprise Designer, double-click **Best Record** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **WeightedCalculator** element.
3. Do either of the following:
    - To modify a custom weighted calculator parameter, scroll to the **candidate-field** element naming the field to modify.
    - To modify a default weighted calculator parameter, scroll to the **default-parameters** element.
4. Modify the value of any of the elements listed in **Table 24 on page 122**.

   For example:

```
<parameter>
    <quality>SourceSystem</quality>
    <preference>DDI</preference>
    <utility>60.0</utility>
</parameter>
```

5 Save and close the file.

Deleting Weighted Calculator Parameters

Once default and custom parameters are defined, they can be deleted if necessary. If a candidate field is defined for custom weighted calculations, you can specify that the field use the default weighted calculator instead by removing the entire field from the **candidate-fields** list.

To delete weighted calculator parameters

1 In the Project Explorer pane of the Enterprise Designer, double-click **Best Record** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **WeightedCalculator** element, and then to the **candidate-field** element identifying the field you want to delete.

3 Do either of the following:

- To delete a custom weighted calculator parameter, scroll to the **candidate-field** element naming the field to modify.
- To delete a default weighted calculator parameter, scroll to the **default-parameters** element.

4 To delete an existing parameter, scroll to the **parameter** element you want to delete, and then remove all text between and including the **parameter** tags for that element.

For example, to delete the SourceSystem parameter below, delete the boldface text.

```
<parameter>
    <quality>SourceSystem</quality>
    <preference>CDI</preference>
    <utility>80.0</utility>
</parameter>
<parameter>
    <quality>MostRecentModified</quality>
    <utility>75.0</utility>
</parameter>
```

Note: *At least one parameter must be defined for the **default-parameter** element; you cannot delete all parameters from this section. You cannot delete all parameters from a candidate field, but you can delete the entire candidate field (see below for more information).*

5 To delete a field from the candidate field list, delete all text between and including the **candidate-field** tags for the field you want to delete.

Using the following example, to delete the **Person.DOB** candidate field from the custom calculator, delete all the text below.

```
<candidate-field name="Person.DOB">
    <parameter>
        <quality>SourceSystem</quality>
        <preference>CDI</preference>
        <utility>80.0</utility>
    </parameter>
    <parameter>
        <quality>MostRecentModified</quality>
        <utility>75.0</utility>
    </parameter>
</candidate-field>
```

6 Save and close the file.

7.5.3 Configuring Update Policies

When the Best Record file is generated, no Java classes are defined for the update policies. You can use standard eView classes, or create custom update policy classes and specify that the custom classes be used instead. Custom update policies must implement **com.stc.eindex.update.UpdatePolicy** and must be defined within **Custom Plug-ins** in the eView Project to be recognized as an update policy. The names of the custom plug-ins you create are the values you enter for the update policies. You can also set the update policy flag to specify whether the policies are performed when no changes are made to an existing record.

Defining Update Policies

You can define update policies for any of the seven update policy elements. You do not need to specify a policy for each element.

To define update policies

1 In the Project Explorer pane of the Enterprise Designer, double-click **Best Record** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **UpdateManagerConfig** section of the file.

3 Do any of the following:

- To modify the merge policy for enterprise objects, change the value of the **EnterpriseMergePolicy** element to the fully qualified name of the new Java class. For example:

```
<EnterpriseMergePolicy>com.stc.eindex.user.MyEntMergePolicy
</EnterpriseMergePolicy>
```

- To modify the unmerge policy for enterprise objects, change the value of the **EnterpriseUnmergePolicy** element to the fully qualified name of the new Java class. For example:

```
<EnterpriseUnmergePolicy>com.stc.eindex.user.MyEntUnmergePolicy
</EnterpriseUnmergePolicy>
```

- To modify the update policy for enterprise objects, change the value of the **EnterpriseUpdatePolicy** element to the fully qualified name of the new Java class. For example:

```
<EnterpriseUpdatePolicy>com.stc.eindex.user.MyEntUpdatePolicy
```

```
</EnterpriseUpdatePolicy>
```

- To modify the create policy for enterprise objects, change the value of the **EnterpriseCreatePolicy** element to the fully qualified name of the new Java class. For example:

```
<EnterpriseCreatePolicy>com.stc.eindex.user.MyCreatePolicy
</EnterpriseCreatePolicy>
```

- To modify the merge policy for system objects, change the value of the **SystemMergePolicy** element to the fully qualified name of the new merge policy Java class. For example:

```
<SystemMergePolicy>com.stc.eindex.user.MySysMergePolicy
</SystemMergePolicy>
```

- To modify the unmerge policy for system objects, change the value of the **SystemUnmergePolicy** element to the fully qualified name of the new Java class. For example:

```
<SystemUnmergePolicy>com.stc.eindex.user.MySysUnmergePolicy
</SystemUnmergePolicy>
```

- To modify the assumed match policy, change the value of the **UndoAssumeMatchPolicy** element to the fully qualified name of the new Java class. For example:

```
<UndoAssumeMatchPolicy>com.stc.eindex.user.MyUndoAsmMatchPolicy
</undoAssumeMatchPolicy>
```

4 Save and close the file.

Setting the Update Policy Flag

The update flag determines whether update policies are performed against a record when a transaction does not cause any changes to the record's data.

To set the update policy flag

1 In the Project Explorer pane of the Enterprise Designer, double-click **Best Record** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **UpdateManagerConfig** section of the file.

3 To specify that update policies are not performed when no updates are made, set the **SkipUpdateIfNoChange** element to "true". For example:

```
<SkipUpdateIfNoChange>true</SkipUpdateIfNoChange>
```

4 To specify that update policies are performed even though no updates are made, set the **SkipUpdateIfNoChange** element to "false". For example:

```
<SkipUpdateIfNoChange>false</SkipUpdateIfNoChange>
```

5 Save and close the file.

Chapter 8

Field Validation Configuration

The Field Validation file defines validations to be performed against certain fields before information is entered into the master index database. This chapter describes the structure of the Field Validation file and provides information about custom field validators.

8.1 Custom Plug-ins

The Field Validation file can be used to associate custom logic with the master index application. The custom logic is created as a Java class using the Custom Plug-ins function of the eView Project in the Enterprise Designer. This chapter provides information you need to know before creating the custom Java classes, including classes to implement, exceptions to call, and so on.

8.2 The Field Validation File

By default, the Field Validation file defines one validation rule named "validate-local-id". This rule defines certain validations that are performed against local ID and system fields before they are entered into the database. The local ID validator verifies that the system code is valid, the local ID format is correct, the local ID is the correct length, and that neither field is null.

8.2.1 Modifying the Field Validation File

You can modify the Field Validation file using the Enterprise Designer XML editor. For more information about this editor, see **"Using the eView Editors" on page 21**. Make sure you are familiar with the procedures described under **"Modifying Configuration Files" on page 24** for checking the file in and out, saving changes, and validating the file. The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes. When you modify this file, you must regenerate the Project and reactivate the deployment for the changes to take effect. You can either apply the changes immediately to the Logical Host, or restart the host to pick up the changes.

8.2.2 Elements

The Field Validation file consists primarily of a list of rules. Each rule is defined within the **ValidationConfig** element and is defined by attributes within a **rules** element. Table 25 describes the elements and attributes of the Field Validation file. A sample appears on the following page.

Table 25 Field Validation Elements

Element or Attribute	Description
rules	Contains a list of validation rules.
rule	Defines a specific validation rule in a **rules** list.
name	A **rule** attribute that specifies a name for the validation rule.
object-name	A **rule** attribute that specifies the name of the class that defines the object to which the validation rule is applied, such as SystemObject or <ParentName>Object (where <ParentName> is the name of the parent object in the Object Definition).
class	A **rule** attribute that specifies the complete path of the Java class containing the validation rule.

8.2.3 Custom Validations

You can define additional validations for your data by creating the Java class and appropriate methods. You must create the custom class using the **Custom Plug-ins** module of the eView Project in order to include your custom validation rule in the master index code package. The custom validation classes must implement **com.stc.eindex.objects.validation.ObjectValidator**. The exception thrown is **com.stc.eindex.objects.validation.exception.ValidationException**.

Plug the custom validation classes into eView by specifying the name of the custom plug-in for the class in the Field Validation file, as shown below.

```
<ValidationConfig module-name="Validation" parser-class=
"com.stc.eindex.configurator.impl.validation.ValidationConfiguration">
     <rules>
          <rule name="validate-auxiliary-id"
               object-name="PersonObject"
               class="com.stc.eindex.user.AuxiliaryId"/>
          <rule name="validate-birth-date"
               object-name="PersonObject"
               class="com.stc.eindex.user.BirthDate"/>
     </rules>
</ValidationConfig>
```

Chapter 9

Enterprise Data Manager Configuration

The Enterprise Data Manager (EDM) is the web-based user interface for the master index that allows you to monitor and modify data in the index. This interface is highly configurable, and can be customized by modifying the Enterprise Data Manager file in the eView Project. This chapter describes the EDM and the Enterprise Data Manager file structure, and provides instructions for configuring the EDM.

For information about how to use the EDM, see the *Enterprise Data Manager User's Guide*.

9.1 About the EDM

The EDM is a web-based interface that allows you to manage and monitor the data in your master index database. Using the EDM, you can search for records; add, update, deactivate, and reactivate records; review and resolve potentially duplicate records; compare records; and merge and unmerge records. You can also view a transaction history for each record and an audit log of access to the database. This interface is highly configurable, allowing you to customize certain processing properties as well as the appearance of certain windows.

9.2 EDM Configuration Components

You can configure several properties of the EDM to display the information you want in the way you want, to define the way searches can be processed, and to define the criteria that can be used for each search. The EDM also defines certain implementation options, such as application or integration server information, debug options, and security information. The configurable properties of the user interface fall into five categories.

- Object and Field Properties
- Relationships
- Display Properties
- Search Page Configuration
- Implementation Configuration

9.2.1 Object and Field Properties

In the Enterprise Data Manager file, you can specify which objects appear on the EDM windows and the order in which they appear. You can also specify the fields displayed in each object. Each field is configured by certain properties. You can specify a field's name, length, order of appearance on the EDM, required data type, whether text can be entered into a field or if it must be selected from a predefined value, whether the a field or combination of fields must be unique to the parent object, and whether the value of the field is hidden under certain circumstances. You can also specify whether the format of a field is dependent on the value of a related field (for example, the format of a credit card number field could be dependent on the type of credit card specified).

9.2.2 Relationships

In the Enterprise Data Manager file, relationships define the hierarchy of the object types listed for the EDM. By specifying relationships, you define parent and children nodes. The parent and child nodes you specify in the **relationships** element must also be defined in the **node** elements of the file. You can specify one parent object; the remaining objects must be child object to the parent you define. This section is dependent on the Object Definition relationships section, and should only be changed if corresponding changes are made to the Object Definition file.

9.2.3 Display Properties

You can configure the appearance of the EDM pages, and whether the audit log is available.

Page Configurations

You can configure several display properties for the pages that appear on the EDM. For most configurable pages, you can specify the number of fields on each row, and for all search pages you can specify the number of results to display on each page. You can also specify the type of object to display on the page, and the name of the tabbed heading. Certain properties of the following pages can be configured.

- Search
- Search Results
- History Search
- History Search Results
- Matching Review Search
- Matching Review Search Results
- View/Edit

Displaying an Audit Log

In the display configuration, you can specify whether an audit log is maintained of all instances in which object information was accessed from the EDM. If the log is maintained, then information about each instance of access can be viewed on the EDM.

9.2.4 Search Page Configuration

Of the configurable pages, the page that requires the most configuration is the Search page. In addition to defining the number of fields per row and the number of records to display in the search results list, you can also specify the search criteria that appear and the types of searches allowed from the EDM.

You can define and name several search pages, each with their own configuration. For each page, you specify groups of fields that are displayed in boxed areas. Each boxed area can represent a different type of search, such as a demographic search, address search, EUID search, and so on.

For each search page you define, you must also specify the search types available, such as alphanumeric or phonetic. You can configure each search type by specifying a name for the search, whether the results are weighted, and whether wildcard characters can be used. When you define the search types for the EDM, you must specify a query for each type you define. The queries you specify must already be defined in the Candidate Select file.

9.2.5 Implementation Configuration

The Enterprise Data Manager file defines certain information about the integration server for the master index implementation, such as the names of certain validation and management components, debug parameters, and security information. The security for the master index is based in the integration server.

9.3 The Enterprise Data Manager File

EDM properties are defined in the Enterprise Data Manager file in XML format. Some of the information entered into the default configuration file is based on the fields you defined in the eView Wizard, and some is standard across all implementations. For most implementations, this file will require customization.

9.3.1 Modifying the Enterprise Data Manager File

You can modify the Enterprise Data Manager file at any time, but you must regenerate the Project and reactivate the deployment after making any changes to the file. You can either apply the changes immediately to the Logical Host, or restart the host to pick up the changes. Changes made to this file do not affect match processing.

Before making any changes to this file, make sure you are familiar with the procedures described under **"Modifying Configuration Files" on page 24**. The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes.

9.3.2 Enterprise Data Manager File Structure

The Enterprise Data Manager file is written in XML, and is automatically generated by the eView Wizard. This file can be modified using the Enterprise Designer XML editor, which is a standard NetBeans editor. This section describes the structure of the XML file and general requirements and constraints. There are five primary components of the Enterprise Data Manager file. The first component, XML File Information, is standard across all implementations.

- XML File Information
- Nodes and Fields
- Relationships
- Implementation Details
- GUI Definition

XML File Information

The first lines in the Enterprise Data Manager file represent information about the schema instance and namespace. You should not need to modify these components.

Nodes and Fields

Each object defined in the object structure has a corresponding **node** element in the Enterprise Data Manager file. Each **node** element is named **node-*object_name***, where *object_name* is the name of the object. Each **node** element contains a list of the fields belonging to that object. Each field is named **field-*field_name***, where *field_name* is the eView name of the field, and each field is configured by a list of elements. For a complete list and description of field elements, see **Table 26 on page 142**.

Most of the information you need to specify in the **node** elements is generated by the eView Wizard when you create the configuration files, but you can modify the information as needed. Below is a sample of a **node** element defining a Phone object that is the third object displayed on the EDM pages. The Phone object contains two fields, PhoneType and PhoneNumber.

```
<node-Phone display-order="3">
    <field-PhoneType>
        <display-name>Phone Type</display-name>
        <display-order>1</display-order>
        <max-length>8</max-length>
        <gui-type>MenuList</gui-type>
        <value-list>PHONTYPE</value-list>
        <value-type>string</value-type>
        <key-type>true</key-type>
    </field-PhoneType>
    <field-Phone>
        <display-name>PhoneNumber</display-name>
        <display-order>2</display-order>
        <max-length>20</max-length>
```

```
                <gui-type>TextBox</gui-type>
                <value-type>string</value-type>
                <input-mask>(DDD)DDD-DDDD</input-mask>
                <value-mask>xDDDxDDDxDDDD</value-mask>
            </field-Phone>
            <is-sensitive>true</is-sensitive>
        </node-Phone>
```

Important: *All fields defined in the Enterprise Data Manager file must also be defined in the Object Definition file; however, not all fields defined in the Object Definition file need to be defined for the EDM. Any fields not listed in the Enterprise Data Manager file will not appear on the EDM.*

Relationships

The **relationships** element in the Enterprise Data Manager file is similar to that of Object Definition. It defines the parent object and child objects. The parent object is identified by a **name** element, and the child objects are identified by **children** elements, as shown below. As with Object Definition, there can only be one parent object but several child objects.

```
<relationships>
    <name>Person</name>
    <children>Address</children>
    <children>Phone</children>
    <children>Alias</children>
</relationships>
```

Implementation Details

The **impl-details** element of the Enterprise Data Manager file defines certain implementation information that is required in order for the EDM to connect to the server. Implementation details also name certain eView components, define debug options, and define security features. There are two primary types of implementation details to configure.

- eView Component Names
- Debug and Security Options

eView Component Names

After the server configuration elements, two elements specify the names of JNDI components of the master index. **master-controller-jndi-name** specifies the JNDI name for the Master Controller component; **validation-service-jndi-name** specifies the JNDI name for the processing code validator, and **usercode-jndi-name** specifies the JNDI name for the validator for user-defined auxiliary IDs. Below is a sample implementation of these elements.

```
<master-controller-jndi-name>ejb/CompanyMasterController
</master-controller-jndi-name>
<validation-service-jndi-name>ejb/CompanyCodeLookup</validation-service-
jndi-name>
<usercode-jndi-name>ejb/CompanyUserCodeLookup</usercode-jndi-name>
```

Debug and Security Options

Following the naming elements in the Enterprise Data Manager file, there are two debug options that control whether debugging is enabled and where the debug information is written. The next implementation options specify whether authorization security is enabled for the EDM and whether the values of certain fields are masked for records of a certain status (such as VIPs or employees). These elements are **debug-flag**, **debug-dest**, **enable-security**, and **object-sensitive-plug-in-class**. If you specify field masking, you must define a custom plug-in to handle the process.

Below is a sample debug and security implementation. These elements are described more fully in **Setting Debug Options** on page 162 and **Configuring Security** on page 163.

```
<debug-flag>true</debug-flag>
<debug-dest>console</debug-dest>
<enable-security>true</enable-security>
<object-sensitive-plug-in-class>
    com.stc.eindex.security.VIPObjectSensitivePlugIn
</object-sensitive-plug-in-class>
```

GUI Definition

You can configure properties of the appearance of the EDM, such as specifying the fields that appear on search windows and results lists, and the number of fields per row on each page. You can also configure search properties, including whether a search is weighted, the fields available for each search, which query types are available, and so on. The EDM configuration elements are all contained in the **page-definition** element in **gui-definition**. The **page-definition** element has five primary elements: **eo-search** configures the standard search page, **create-eo** configures the Create System Record page, **history** configures the Transaction History pages, **matching-review** configures the Match Review pages, and **audit-log** specifies whether audit logging is enabled.

eo-search

The **eo-search** element defines the configuration of the search and search results pages. It contains five primary elements: **root-object**, **tab-name**, **tab-entrance**, **simple-search-page**, **search-result-list-page**, and **eo-view-page**. Each element can only occur once in this section except **simple-search-page**, which can have multiple occurrences. The first three elements define the type of object returned by a search, the name of the search page, and the URL to the search page (this is used internally). The last three elements are described below.

- **simple-search-page—**Each **simple-search-page** element defines a search page, and consists of many elements that specify a name for the search type, the appearance of the page, the possible criteria for the search, and the types of searches available on the page. The following elements can be defined for a search page: **screen-title**, **field-per-row**, **show-euid**, **show-lid**, **instruction**, **field-group**, and **search-option**. These elements are described in **Table 28 on page 151**.

  Each **field-group** element represents a boxed area on the search page and contains a list of **field-ref** elements that define the fields that appear in each box. These elements are described in **Table 29 on page 152**.

Each **search-option** element includes several elements that define the properties of each search, such as which query from Candidate Select to use for the search type, whether the search is weighted, and so on. If one **simple-search-page** element includes several **search-option** elements, the names of the search options appear at the top of the search page next to option buttons. EDM users can select an option button to specify the type of search to perform on that page. Search option elements are described in **Table 30 on page 153**.

A sample of the **simple-search-page** section of the Enterprise Data Manager file appears below. This sample defines one simple search page with a search type names "Advanced Person Lookup" and with two option buttons ("Alphanumeric" and "Blocking").

```
<simple-search-page>
    <screen-title>Advanced Person Lookup</screen-title>
    <field-per-row>2</field-per-row>
    <show-euid>false</show-euid>
    <show-lid>false</show-lid>
    <instruction>Enter known values in the fields below</instruction>
    <field-group>
        <description>Name</description>
        <field-ref>Person.LastName</field-ref>
        <field-ref>Person.FirstName</field-ref>
        <field-ref>Person.MiddleName</field-ref>
    </field-group>
    <field-group>
        <description>Address</description>
        <field-ref>Address.AddressType</field-ref>
        <field-ref>Address.AddressLine1</field-ref>
        <field-ref>Address.AddressLine2</field-ref>
    </field-group>
    <search-option>
        <display-name>Alphanumeric</display-name>
        <query-builder>ALPHA-SEARCH</query-builder>
        <weighted>false</weighted>
        <parameter>
            <name>UseWildcard</name>
            <value>true</value>
        </parameter>
    </search-option>
    <search-option>
        <display-name>Blocking</display-name>
        <query-builder>BLOCKER-SEARCH</query-builder>
        <weighted>true</weighted>
        <parameter>
            <name>UseWildCard</name>
            <value>false</value>
        </parameter>
    </search-option>
</simple-search-page>
```

Note: *To include the EUID and local ID on a search page in their own labelled boxes, you do not need to define **field-group** elements for them. Simply set **show-euid** and **show-lid** to **true**, and the EUID field appears in one labelled box, and the system and local ID fields appear in another.*

- **search-result-list-page**—This element specifies the fields in the search results list, the maximum number of records to return from a search, and the number of results to display at one time. It contains one **item-per-page** and one **max-result-size** element and several **field-ref** elements. For example:

```
<search-result-list-page>
    <item-per-page>10</item-per-page>
    <max-result-size>500</max-result-size>
    <field-ref>Person.LastName</field-ref>
    <field-ref>Person.FirstName</field-ref>
    <field-ref>Person.MiddleName</field-ref>
    <field-ref>Person.DOB</field-ref>
    <field-ref>Person.Gender</field-ref>
    <field-ref>Person.SSN</field-ref>
    <field-ref>Address.AddressLine1</field-ref>
</search-result-list-page>
```

The EUID is always included in the results list, though it is not listed in this section.

- **eo-view-page**—This element defines the number of fields contained in each row of the View/Edit page. Unless the objects in the data structure contain very few fields, SeeBeyond recommends leaving this value at “1” for readability. For example:

```
<eo-view-page>
    <field-per-row>1</field-per-row>
</eo-view-page>
```

create-eo

The **create-eo** element defines the configuration of the EDM page that allows users to enter new object records. It contains three elements. The **root-object** element defines the name of the object displayed on the page. The **tab-name** element specifies a name for the tabbed heading that, when clicked, opens the Create System Record page. The **tab-entrance** element specifies the URL for entry into the CreateEO page (the URL is used internally and should not be modified). Below is a sample of the **create-eo** element.

```
<create-eo>
    <root-object>Person</root-object>
    <tab-name>Create System Record</tab-name>
    <tab-entrance>/stcedm/EnterEOCreateAction.do</tab-entrance>
</create-eo>
```

history

The **history** element defines the configuration of the History page, which displays a history of changes to an object’s record. This element contains the same three elements as described for the **create-eo** element above. It also contains an **xa-search-page** element, a **search-result-list-page** element, and a **merge-history-key-field** element.

The **xa-search-page-element** contains one element, **field-per-row**, which specifies the number of fields to display on each row of the History Search page.

The **search-result-list-page** element contains three types of elements. The **item-per-page** element specifies the number of records that appear on each History Search Result page. The **max-result-size** element specifies the maximum number of records that can be returned from a History search. A list of **field-ref** elements defines fields that appear on the History Search Result page in addition to those that are permanent. Permanent fields in the results list include TransactionID, EUID1, EUID2, System, LID1, LID2, Function, SystemUser, and TimeStamp.

The **merge-history-key-field** element contains a list of **field-ref** elements that define the fields that appear on the merge tree (in addition to those that permanently appear in the merge tree transaction table). Permanent fields in the transaction table include TransactionID, EUID1, EUID2, System, LID1, LID2, Function, SystemUser, and TimeStamp. Any fields specified in this list will also appear on the History Search Result page, but if a field is listed in both this element and the **search-result-list-page** element, only one instance of the field appears in the results list.

Below is a sample of the History page configuration.

```
<history>
    <root-object>Person</root-object>
    <tab-name>History</tab-name>
    <tab-entrance>/stcedm/EnterXASearchAction.do</tab-entrance>
    <xa-search-page>
        <field-per-row>2</field-per-row>
    </xa-search-page>
    <search-result-list-page>
        <item-per-page>10</item-per-page>
        <max-result-size>500</max-result-size>
        <field-ref>Person.FirstName</field-ref>
        <field-ref>Person.LastName</field-ref>
        <field-ref>Phone.Phone</field-ref>
        <field-ref>Address.AddressLine1</field-ref>
    </search-result-list-page>
    <merge-history-key-field>
        <field-ref>Person.FirstName</field-ref>
        <field-ref>Person.LastName</field-ref>
    </merge-history-key-field>
</history>
```

matching-review

The **matching-review** element defines the configuration of the Matching Review page, which displays two records side-by-side in a comparison view. This element contains the same three elements as described for the **create-eo** element above. It also contains a **pd-search-page** element and a **search-result-list-page** element.

The **pd-search-page-element** contains one element, **field-per-row**, which specifies the number of fields to display on each row of the matching review search page.

The **search-result-list-page** element contains an **item-per-page** element that specifies the number of records to display on each Matching Review Search Result page, and a **max-result-size** element that specifies the maximum number of records to return for a matching review search. You cannot define additional fields to display on this page.

Below is a sample of the Matching Review page configuration.

```
<matching-review>
    <root-object>Person</root-object>
    <tab-name>Matching Review</tab-name>
    <tab-entrance>/stcedm/EnterPDSearchAction.do</tab-entrance>
    <pd-search-page>
        <field-per-row>2</field-per-row>
    </pd-search-page>
    <search-result-list-page>
        <item-per-page>15</item-per-page>
        <max-result-size>500</max-result-size>
    </search-result-list-page>
</matching-review>
```

audit-log

The **audit-log** element specifies whether an audit log is maintained of all instances in which enterprise object information was accessed from the master index database. If the audit log is maintained, the Audit Log page is available on the EDM for searching and viewing audit log entries.

Below is a sample of the audit log configuration.

```
<audit-log>
    <allow-insert>true</allow-insert>
</audit-log>
```

9.4 Customizing the Enterprise Data Manager File

The EDM is highly configurable, and you can specify information about the appearance of the EDM, the order and appearance of fields, which search types are available, the available search criteria, and so on. Configuring the EDM consists of four primary components.

- **Configuring Fields and Objects** on page 138
- **Configuring the Search Pages** on page 149
- **Defining Page Layouts** on page 156
- **Configuring Implementation Information** on page 161

All of these actions require that you use the Enterprise Designer XML editor. For more information about this editor, see **"Using the eView Editors" on page 21**. Make sure you are familiar with the procedures described under **"Modifying Configuration Files" on page 24** for checking the file in and out, saving changes, and validating the file.

9.5 Configuring Fields and Objects

You can modify the configuration of the fields and objects EDM to specify how they appear on the EDM pages. You can perform any of the following actions to customize the general appearance of the EDM.

- **Modifying Object Names** on page 139
- **Adding an Object** on page 139
- **Deleting Objects** on page 140
- **Configuring Fields** on page 141
- **Defining Relationships** on page 148

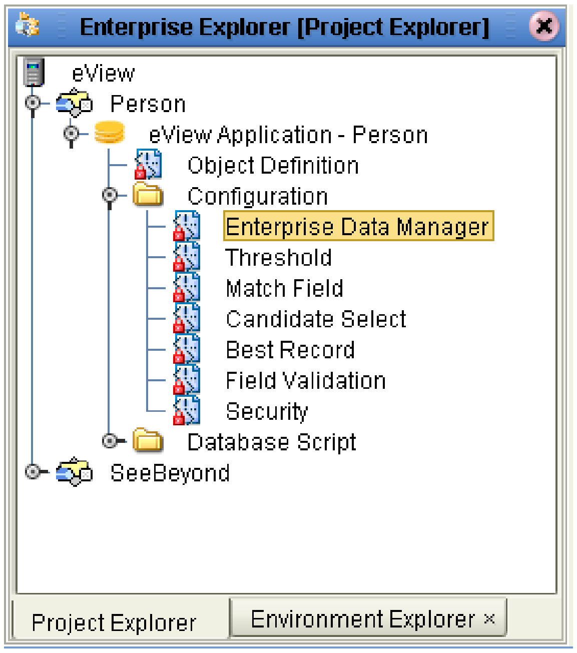
9.5.1 Modifying Object Names

Once an object is defined in the Enterprise Data Manager file, you can modify the name if necessary. **Only modify the name of an object if you modify the corresponding object name in Object Definition and the remaining configuration files.** It is not recommended that the parent object name be changed.

To modify an object name

1 In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.

Figure 7 Enterprise Data Manager Node



2 Scroll to the **node** element naming the object to modify.

3 Change the value of the **node** element to **node-*object_name***, where *object_name* is the new name for the object.

```
<edm xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="EDM.xsd">
    <node-Customer>
```

4 Save and close the file.

9.5.2 Adding an Object

You can define additional objects for the EDM as long as those objects are defined in the Object Definition. Each object can only contain the fields that are also defined for that object in the Object Definition.

To add an object name

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. In the nodes list of the file, create a new **node-*object*** element, substituting the object name for ***object***, as shown below.

```
<node-Phone>
</node-Phone>
```

3. For any child objects, specify the display order of the object in the **display-order** attribute in the **node-*object*** tag. For example:

```
<node-Phone display-order="3">
</node-Phone>
```

4. Define fields for the new object, as described in **"Defining a New Field" on page 141**.
5. Define the relationship of the object, as described in **"Defining Relationships" on page 148**.
6. Save and close the file.

9.5.3 Deleting Objects

Once an object is defined in the Enterprise Data Manager file, you can delete the object if necessary. If the object remains defined in the Object Definition file, then the object is still a part of the enterprise record, but does not appear on the EDM.

To delete an object

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **node** element naming the object to delete.
3. Delete all text between the start and ending **node** tags for that object.

   Using the sample below, to delete the Phone object, delete all the text in the sample.

```
<node-Phone display-order="3">
    <field-PhoneType>
        <display-name>Phone Type</display-name>
        <display-order>1</display-order>
        <max-length>8</max-length>
        <gui-type>MenuList</gui-type>
        <value-list>PHONTYPE</value-list>
        <value-type>string</value-type>
        <key-type>true</key-type>
    </field-PhoneType>
    <field-Phone>
        <display-name>Phone</display-name>
        <display-order>2</display-order>
        <max-length>20</max-length>
        <gui-type>TextBox</gui-type>
        <value-type>string</value-type>
        <input-mask>(DDD)DDD-DDDD</input-mask>
        <value-mask>xDDDxDDDxDDDD</value-mask>
    </field-Phone>
</node-Phone>
```

4 If necessary, renumber the order of the remaining objects so they are sequential.

5 Delete the relationship of the object, as described in **"Defining Relationships" on page 148**.

6 Save and close the file.

9.5.4 Configuring Fields

There are a variety of field properties that define how and where a field appears on the EDM, what type of data a field can accept, the length of the field, and so on. To define field properties, perform any of the following actions.

- **Defining a New Field** on page 141
- **Modifying a Field's Display Name** on page 143
- **Modifying a Field's Location on the EDM** on page 144
- **Modifying a Field's Length** on page 144
- **Modifying a Field's Display Type** on page 145
- **Specifying a Drop-down List for a Field** on page 145
- **Specifying a Field Display Format** on page 146
- **Modifying the Data Type for a Field** on page 146
- **Modifying a Field's Key Status** on page 146
- **Masking Field Values on the EDM** on page 147

Defining a New Field

You can define new fields for an object in the Enterprise Data Manager file, but the field must correspond with a field defined for that object in Object Definition. The fields defined here appear on the EDM windows.

To define new fields

1 In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **node** element that defines the object to which you want to add a field.

3 In the **node** element, create a new element named **field-*field_name***, where *field_name* is the name of the field as it appears in Object Definition file. For example:

```
<node-Person>
     <field-LastName>
     </field-LastName>
</node>
```

4 Define and configure the elements listed in Table 26 for the new **field-*field_name*** element. For example:

```
<field-LastName>
    <display-name>LastName</display-name>
    <display-order>2</display-order>
    <max-length>40</max-length>
    <gui-type>TextBox</gui-type>
    <value-type>string</value-type>
    <key-type>false</key-type>
</field-LastName>
```

5 Save and close the file.

Table 26 EDM Field Configuration Elements

Element	Description
display-name	The name of the field as it will appear on the EDM.
display-order	The order in which the field appears on the EDM. For example, specify **1** to indicate this is the first field on the EDM pages, **2** to indicate it is the second field, and so on.
max-length	The maximum number of characters displayed on the EDM for the field.
gui-type	An indicator of the type of display for the field. Specify **TextBox** for a standard data entry field; **MenuList** for a field that must be populated by selecting from a drop-down list; or **TextArea** for a long field that requires a scrollbar, such as a comments field.
value-type	The eView data type for the data populated in the field.
input-mask	A mask used by the EDM to add punctuation to a field. You can add an input mask to display telephone numbers as "(123)456-7890 even though no punctuation is entered. 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 telephone format is "(DDD)DDD-DDDD". The following character types are allowed: ▪ D - indicates a numeric character. ▪ L - indicates an alphabetic character. ▪ A - indicates an alphanumeric character. ***Note:*** *If the length of the input mask is greater than the value specified for the max-length element, the length of the input mask is used.*

Table 26 EDM Field Configuration Elements

Element	Description
value-mask	A mask used by the index to strip any extra characters that were added by the input mask. This mask ensures that data is stored in the database in the correct format. This mask must be the same length as the input mask. To specify a value mask, type the same value as is entered for the input mask, but type an "x" in place of each punctuation mark. For example, if an SSN field has an input mask of DDD-DD-DDDD, you need to specify a value mask of DDDxDDxDDDD to strip the dashes before storing the SSN. A value mask is not required for date fields.
value-list	The name of the menu list used to populate the drop-down list for the field. This is required if the **gui-type** specified is "MenuList", and must match a code of an element in the sbyn_common_header database table.
key-type	An indicator of whether the field (or a combination of key fields) must be unique in an enterprise record. Unique key fields identify unique child objects in an enterprise object. Specify **true** to indicate the field is a key field; specify **false** if it is not.
is-sensitive	An indicator of whether the value of the field is hidden on the EDM for records with a VIP status of "VIP". Only users with the eView.Admin or eView.VIP user roles can view the hidden information. Specify **true** to hide the field value; specify **false** (or remove the **is-sensitive** element) to display the field value. ***Note:*** *This element is only used if the* ***object-sensitive-plug-in-class*** *in the* ***impl-details*** *section is populated.*

Modifying a Field's Display Name

Once a field is defined for an object in the Enterprise Data Manager file, you can change the name that appears on the EDM for that field.

To modify a field's display name

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **node** element that defines the object that contains the field you want to modify.
3. Scroll to the **field-*field_name*** element you want to modify, where *field_name* is the name of the field as it appears in Object Definition.

4. Change the value of the **display-name** element. For example:

   ```
   <display-name>Last Name</display-name>
   ```

5. Save and close the file.

Modifying a Field's Location on the EDM

Once the fields are defined for each object in the Enterprise Data Manager file, you can modify the order of the fields as they appear on the EDM windows.

To modify a field's location on the EDM windows

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **node** element that defines the object containing the field you want to modify.
3. Scroll to the **field-*field_name*** element you want to modify, where *field_name* is the name of the field as it appears in Object Definition.
4. Change the value of the **display-order** element. For example:

   ```
   <display-order>1</display-order>
   ```

Note: *If you change the order of one field, you must change the order of at least one other field to maintain sequential numbering. For example, if you change a field's location from "2" to "1", you must then change the location of the field originally specified for location 1.*

5. Save and close the file.

Modifying a Field's Length

Once a field is defined for an object in the Enterprise Data Manager file, you can change the number of characters you can enter for the field in the EDM.

Important: *This length is constrained by the length of the database column containing this field and the length defined in the Object Definition file.*

To modify a field's length

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **node** element that defines the object containing the field you want to modify.
3. Scroll to the **field-*field_name*** element you want to modify, where *field_name* is the name of the field as it appears in Object Definition.
4. Change the value of the **max-length** element. For example:

   ```
   <max-length>100</max-length>
   ```

5. Save and close the file.

Modifying a Field's Display Type

Once a field is defined for an object in the Enterprise Data Manager file, you can change the type of field displayed on the EDM.

To modify a field type

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **node** element that defines the object containing the field you want to modify.
3. Scroll to the **field-*field_name*** element you want to modify, where *field_name* is the name of the field as it appears in Object Definition.
4. Change the value of the **gui-type** element. For example:

```
<gui-type>TextBox</gui-type>
```

Note: *Two values are allowed: "TextBox", for a standard field, and "MenuList", for a field whose value must be selected from a drop-down list.*

5. Save and close the file.

Specifying a Drop-down List for a Field

Once a field is defined for an object in the Enterprise Data Manager file, you can specify or change the name of the drop-down list for the field.

To modify a field type

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **node** element that defines the object containing the field you want to modify.
3. Scroll to the **field-*field_name*** element you want to modify, where *field_name* is the name of the field as it appears in Object Definition.
4. Change the value of the **value-list** element.

   If the element does not exist for the field, create a new **value-list** element. For example:

```
<value-list>State</value-list>
```

Note: *The value of the **gui-type** element for the field must be "MenuList" if you specify a drop-down list. The **value-list** element must match a code column value in the sbyn_common_header database table unless the drop-down list is populated by information in the sbyn_user_code table (as they might be for auxiliary IDs). In this case, the **value-list** element must match a code_list column value in sbyn_user_code.*

5. Save and close the file.

Specifying a Field Display Format

Once a field is defined, you can specify or modify a display format for the field. This automatically enters punctuation into a field on the EDM, but removes the punctuation in the database.

To modify a field type

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **node** element that defines the object containing the field you want to modify.
3. Scroll to the **field-*field_name*** element you want to modify, where *field_name* is the name of the field as it appears in Object Definition.
4. Change the value of the **input-mask** and **value-mask** elements.

   If the element does not exist for the field, create new **input-mask** and **value-mask** elements. For example:

   ```
   <input-mask>(DDD)DDD-DDDD</input-mask>
   <value-mask>xDDDxDDDxDDDD</value-mask>
   ```

Note: *If an input mask is defined, in most cases a value mask must also be defined.*

5. Save and close the file.

Modifying the Data Type for a Field

Each field on the EDM requires a specific type of data to be entered. For example, name fields generally require a data string and date fields require a valid date or numeric characters. The type of data defined for each field must correspond with the field type defined for that field in the Object Definition file and in the database.

To modify the data type

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **node** element that defines the object that contains the field you want to modify.
3. Scroll to the **field-*field_name*** element you want to modify, where *field_name* is the name of the field as it appears in Object Definition.
4. Change the value of the **value-type** element. For example:

   ```
   <value-type>string</value-type>
   ```

5. Save and close the file.

Modifying a Field's Key Status

You can specify that a certain field or combination of fields be unique to the parent object. An example of a unique fields would be the address type if only one address of each type is allowed. A field's key type status in the Enterprise Data Manager file must match its key type status in the Object Definition file.

To modify the key status

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **node** element that defines the object containing the field you want to modify.
3. Scroll to the **field-*field_name*** element you want to modify, where *field_name* is the name of the field as it appears in Object Definition.
4. Change the value of the **key-type** element.

   If the element does not exist for the field, create a new **key-type** element. For example:

   ```
   <key-type>false</key-type>
   ```

   Specify **true** if the field must be unique; specify **false** if it does not need to be unique.

Note: *You can specify that a combination of fields be unique rather than just a single field.*

5. Save and close the file.

Masking Field Values on the EDM

You can specify that the values of certain fields be hidden on the EDM from users without the appropriate access permissions. Field values are hidden for records with a VIP status of "VIP"; in all other records the field values are visible regardless of whether the field is marked for masking. This option is only available if the **object-sensitive-plug-in-class** element in the **impl-details** section is populated.

Important: *To mask field values, you must define a custom plug-in to implement the masking rules. This class is specified in the* ***object-sensitive-plug-in-class*** *element.*

To mask field values on the EDM

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **node** element that defines the object containing the field you want to modify.
3. Scroll to the **field-*field_name*** element you want to modify, where *field_name* is the name of the field as it appears in Object Definition.
4. Change the value of the **is-sensitive** element to "true".

   If the element does not exist for the field, create a new **is-sensitive** element. For example:

   ```
   <is-sensitive>true</is-sensitive>
   ```
5. Save and close the file.

9.5.5 Deleting a Field from an Object

If a field is defined for an object in the Enterprise Data Manager file, that field appears on the EDM windows. You can remove the field from the EDM by deleting the field definition.

To delete a field definition

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **node** element that defines the object from which you want to delete a field.
3. In the **node** element, scroll to the **field-*field_name*** element you want to delete, where *field_name* is the name of the field.
4. Delete all text between and including the **field-*field_name*** element.

   For example, to delete the Nationality field, delete all text in the sample below.

```
<field-Nationality>
    <display-name>Nationality</display-name>
    <display-order>12</display-order>
    <max-length>8</max-length>
    <gui-type>MenuList</gui-type>
    <value-type>string</value-type>
    <key-type>false</key-type>
</field-Nationality>
```

5. If necessary, renumber the remaining fields, as described in **“Modifying a Field's Location on the EDM” on page 144**.
6. Save and close the file.

9.5.6 Defining Relationships

The relationships in the Enterprise Data Manager file are predefined based on the information you provided when you created the object structure definition. The relationship structure in the Enterprise Data Manager file should match that of the Object Definition.

To define relationships

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **relationships** element.
3. Specify the name of the parent object in the **name** element.
4. Specify the name of the children objects in the **children** elements. For example:

```
<relationships>
    <name>Person</name>
    <children>Alias</children>
    <children>Address</children>
    <children>Phone</children>
    <children>AuxId</children>
</relationships>
```

5 To remove a child object from the relationships list, delete all text between and including the **children** tags defining the object you want to delete.

6 Save and close the file.

9.6 Configuring the Search Pages

The **eo-search** element, which is nested in the **page-definition** element of the **gui-definition** element, contains all of the configuration information for the search pages that appear on the EDM. You can perform any of the following actions to configure the search pages of the EDM.

- **Specifying Standard Search Page Properties** on page 149
- **Creating a Search Page** on page 150
- **Modifying Search Pages** on page 154
- **Defining Page Layouts** on page 156

9.6.1 Specifying Standard Search Page Properties

Standard search properties include the type of object returned by each search, the name of the tabbed header for the search pages, and the URL for entry into the search area. These properties apply to all search pages you define, and they can be modified as needed.

To specify standard search page properties

1 In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **page-definition** element in the **gui-definition** element.

3 In the **eo-search** element, modify any of the elements defined in Table 27 except the **tab-entrance** element.

```
<root-object>Company</root-object>
<tab-name>Company Search</tab-name>
<tab-entrance>/stcedm/EnterEOSearchSimpleAction.do</tab-entrance>
```

4 Save and close the file.

Table 27 Standard Search Elements

Element	Description
root-object	The name of the type of object returned by the search (this must be the parent object).
tab-name	A name for the search pages. This name appears on tab label associated with the search pages on the EDM.
tab-entrance	The URL to the entry page of the search pages. This element should not be modified.

9.6.2 Creating a Search Page

Several search pages are defined by the eView Wizard, including the simple lookup page, advanced lookup pages, and the comparison lookup page. You can add additional search pages if needed. Perform the following steps to create a new search page.

- **Step 1: Define the Search Page** on page 150
- **Step 2: Define the Search Fields** on page 151
- **Step 3: Specify Search Options** on page 152

Step 1: Define the Search Page

The first step in creating a search page is to define certain properties for the appearance of the page, such as its name, how many fields to list in each row, whether to display the EUID or local ID field, and general instructions for the search.

Important: *If either the EUID field or the local ID and system fields appear on a search page, any values entered into these optional fields take precedence over information entered into other search fields. For example, if an invalid EUID is entered but valid first and last names are entered, no results are returned due to the invalid EUID. The EUID field takes precedence over the local ID and system fields.*

To define the search page

1 In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **page-definition** element in the **gui-definition** element.

3 In the **eo-search** element, create a **simple-search-page** element.

Make sure the new element falls within the **eo-search** element, but outside any existing **simple-search-page** elements. For example:

```
<eo-search>
    <simple-search-page>
     ...
    </simple-search-page>
    <simple-search-page>
    </simple-search-page>
</eo-search>
```

4 In the new **simple-search-page** element, create the elements listed in **Table 28 on page 151** and enter the appropriate value for each element. For example:

```
<eo-search>
    <simple-search-page>
     ...
    </simple-search-page>
    <simple-search-page>
        <screen-title>Address Search</screen-title>
        <field-per-row>1</field-per-row>
        <show-euid>true</show-euid>
        <show-lid>false</show-lid>
        <instruction>Enter address information below.</instruction>
    </simple-search-page>
```

```
</eo-search>
```

5 Continue to **"Step 2: Define the Search Fields"**.

Table 28 EDM Search Page Definition Elements

Element	Description
screen-title	The name of the search as it appears in the Search Type drop-down list, from which users can select a type of search to perform.
field-per-row	The number of fields to display in each row on the search page.
show-euid	An indicator of whether to display the EUID. Specify **true** to display the EUID; otherwise specify **false**. Only display this field if you want it to take precedence over all other search criteria. When the EUID is displayed, it appears in its own labelled box.
show-lid	An indicator of whether to display the local ID and system fields. Specify **true** to display the fields; otherwise specify **false**. Only display these fields if you want them to take precedence over all other search criteria (except the EUID field). When the local ID is displayed, the local ID and system fields appear in their own labelled box.
instruction	A short statement to help the user process a search. The text you enter here appears above the search fields on the Search page.

Step 2: Define the Search Fields

Once you define the search page, you must specify the fields that appear on the page. Fields are specified in field groups, and each field group represents a boxed area on the search page. All fields specified for a field group appear in the boxed area defined by that group. The box label is defined by the description of the field group.

To define search fields

1 Complete **"Step 1: Define the Search Page"**.

2 In the new **simple-search-page** element, create a **field-group** element. For example:

```
<simple-search-page>
    <screen-title>Simple Person Search</screen-title>
    <field-per-row>2</field-per-row>
    <show-euid>false</show-euid>
    <show-lid>false</show-lid>
    <field-group>
    </field-group>
</simple-search-page>
```

3 In the new **field-group** element, create the elements listed in **Table 29 on page 152** and enter the appropriate value for each element. For example:

```
<simple-search-page>
    <screen-title>Simple Person Search</screen-title>
    <field-per-row>2</field-per-row>
```

```
        <show-euid>false</show-euid>
        <show-lid>false</show-lid>
        <field-group>
            <description>Address</description>
            <field-ref>Address.AddressType</field-ref>
            <field-ref>Address.AddressLine1</field-ref>
            <field-ref>Address.AddressLine2</field-ref>
            <field-ref>Address.City</field-ref>
            <field-ref>Address.State</field-ref>
        </field-group>
    </simple-search-page>
```

4 Repeat steps 2 and 3 for each field group you want to display on the selected search page.

5 Continue to **"Step 3: Specify Search Options"**.

Table 29 EDM Field Group Elements for a Search

Element	Description
description	A description of the fields defined for the **field-group** element. This value appears as a box label for the area of the page that contains the specified fields.
field-ref	The simple field names of the fields in the field group using their corresponding objects as the root. For example, that path to the FirstName field in the Person object is "Person.FirstName". You can define multiple **field-ref** elements for each field group.

Step 3: Specify Search Options

After you define the criteria fields for the EDM search, you must specify certain options for the search, such as the types of available searches, whether each search is weighted, and whether the search allows wildcard characters.

To specify search options

1 In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.

2 In the new **simple-search-page** element, create a **search-option** element. For example:

```
    <simple-search-page>
        <screen-title>Simple Person Search</screen-title>
        <field-per-row>2</field-per-row>
        <show-euid>false</show-euid>
        <show-lid>false</show-lid>
        <field-group>
            ...
        </field-group>
        <search-option>
        </search-option>
    </simple-search-page>
```

3 In the new **search-option** element, create the elements listed in **Table 30 on page 153** and enter the appropriate value for each element. For example:

```
<search-option>
    <display-name>Alpha Search</display-name>
    <query-builder>ALPHA-SEARCH</query-builder>
    <weighted>false</weighted>
    <parameter>
    </parameter>
</search-option>
```

4 If you specified a **parameter** element, create the elements listed in **Table 31 on page 153** within that option, and enter the appropriate values for each element. For example:

```
<parameter>
    <name>UseWildCard</name>
    <value>true</value>
</parameter>
```

5 Repeat steps 2 through 4 for each search type you want to make available on the selected search page.

Note: *If you define multiple search option elements, an option button (labelled by the value of the* ***display-name*** *element) appears on the search page for each search option.*

6 Save and close the file.

Table 30 EDM Search Option Elements

Element	Description
display-name	A short phrase describing the type of search to perform, such as “Alphanumeric Search” or “Phonetic Search”. This appears next to the option button on the search page when multiple search options are defined.
query-builder	The type of query to use when this type of search is selected. The value entered here must match a **query-builder** name in the Candidate Select file.
weighted	An indicator of whether the results of the search are assigned matching probability weights. Specify **true** to assign matching weights; specify **false** to return unweighted results.
parameter	A list of optional parameters for the search.

Table 31 EDM Search Parameter Elements

Element	Description
name	The name of the parameter. Currently, only **UseWildCard** is available.
value	The value of the parameter. For the **UseWildCard** parameter, this is an indicator of whether the parameter is enabled or disabled. Specify **true** to allow wildcard characters; specify **false** to perform exact-match searches.

9.6.3 Modifying Search Pages

Once a search page is defined, it can be modified as needed. You can perform any of the following actions to customize existing search page elements.

- **Modifying a Search Page Definition** on page 154
- **Modifying Search Fields** on page 154
- **Modifying Search Options** on page 155

Modifying a Search Page Definition

Once a search page is defined in the Enterprise Data Manager file, you can modify the search page definition. The following properties can be modified: the name of the search, the number of fields that appear on each row of the search page, and whether the EUID or local ID fields are visible.

To modify a search page definition

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **page-definition** element in the **gui-definition** element.
3. Scroll to the **eo-search** element, and then to the **simple-search-page** element you want to modify.
4. In the **simple-search-page** element, change the value of any of the elements listed in **Table 28 on page 151**. For example:

```
<simple-search-page>
    <screen-title>Customer Search</screen-title>
    <field-per-row>2</field-per-row>
    <show-euid>true</show-euid>
    <show-lid>false</show-lid>
    <instruction>Enter the EUID below.</instruction>
</simple-search-page>
```

5. Save and close the file.

Modifying Search Fields

Once field groups and fields are specified for a defined search page, you can modify the properties of the group and of the fields contained in a group.

To modify search fields

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **page-definition** element in the **gui-definition** element.
3. Scroll to the **eo-search** element, and then to the **simple-search-page** element you want to modify.
4. In the new **simple-search-page** element, scroll to the **field-group** you want to modify, and do any of the following:

- To modify the name of the boxed area in which the field group appears in the EDM, change the value of the **description** element.
- To add a new field to a field group, create and name a new **field-ref** element in the appropriate **field-group** element.
- To modify the name of a field defined for a field group, change the value of the appropriate **field-ref** element.
- To delete a field from a field group, delete all text between and including the **field-ref** tags that define the field to be deleted.
- To delete an entire field group, delete all text between and including the **field-group** tags that define the field group to be deleted.

5 Save and close the file.

Modifying Search Options

Once search options are defined for a search page, you can modify these options if needed.

To modify search options

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **page-definition** element in the **gui-definition** element.
3. Scroll to the **eo-search** element, and then to the **simple-search-page** element you want to modify.
4. In the new **simple-search-page** element, scroll to the **search-option** element, and do any of the following:
   - To modify the name of the search option button, change the value of the **display-name** element.
   - To modify the query type of the selected search, change the value of the **query-builder** element. The query you specify must match a query defined in the Candidate Select file.
   - To specify that a search return weighted results, change the value of the **weighted** element to **true**.
   - To specify that a search return unweighted results, change the value of the **weighted** element to **false**.
   - To specify that wildcard characters can be used in a search, change the **UseWildCard** parameter **value** element to **true**.
   - To specify that wildcard characters cannot be used in a search, change the **UseWildCard** parameter **value** element to **false**.
5. Save and close the file.

9.7 Defining Page Layouts

You can define how fields appear on most EDM pages. You can also define the names of the tabbed heading for a page and the type of object handled on each page. Perform any of the following actions to define screen layouts.

- **Configuring the Search Results Page** on page 156
- **Configuring the View/Edit Page** on page 157
- **Configuring the Create System Record Page** on page 157
- **Configuring the History Page** on page 158
- **Configuring the Match Review Page** on page 159
- **Configuring the Audit Log Pages** on page 160

9.7.1 Configuring the Search Results Page

The Search Results page displays a list of records returned from a search. The same Search Results page appears for all simple search pages. You can configure the number of records to display at one time in the results list and the fields to display in the list. Fields in the **field-ref** elements are named by their object name and then the field name; for example, Person.LastName or Phone.PhoneNumber.

To configure the Search Results page

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **page-definition** element in the **gui-definition** element, and then to the **eo-search** element.
3. In the **search-result-list-page** element, do any of the following:

- To modify the number of records displayed on the search results page at one time, change the value of the **item-per-page** element. For example:

```
<item-per-page>15</item-per-page>
```

- To specify a maximum number of records to return from a search, change the value of the **max-result-size** element. For example:

```
<max-result-size>100</max-result-size>
```

- To add a new field to the results list, create and name a new **field-ref** element.

```
<field-ref>Person.LastName</field-ref>
```

- To modify a field in the results list, change the value of the appropriate **field-ref** element. For example:

```
<field-ref>Person.FirstName</field-ref>
```

- To delete a field from the results list, delete all text between and including the **field-ref** tags defining the field to be deleted.

  For example, to delete the City field from the following list, delete the boldface text.

```
<search-result-list-page>
    <item-per-page>10</item-per-page>
    <field-ref>Company.CompanyName</field-ref>
    <field-ref>Company.Industry</field-ref>
    <field-ref>Address.AddressLine1</field-ref>
    <field-ref>Address.City</field-ref>
</search-result-list-page>
```

4 Save and close the file.

9.7.2 Configuring the View/Edit Page

The View/Edit page appears when you select a record in the search results list. This page displays all of the information about an enterprise object. You can configure the number of fields that appear in each row of the view page.

To configure the View page

1 In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **page-definition** element in the **gui-definition** element, and then to the **eo-search** element.

3 In the **eo-view-page** element, change the value of the **field-per-row** element. For example:

```
<eo-view-page>
    <field-per-row>2</field-per-row>
</eo-view-page>
```

4 Save and close the file.

9.7.3 Configuring the Create System Record Page

The Create System Record page is where a user adds new records to the master index database. You can configure the name of the tabbed heading and the type of object returned. Do not modify the URL of the entrance to the Create System Record page. Following is a sample of the **create-eo** element.

```
<create-eo>
    <root-object>Person</root-object>
    <tab-name>Create System Record</tab-name>
    <tab-entrance>/stcedm/EnterEOCreateAction.do</tab-entrance>
</create-eo>
```

To configure the Create EO page

1 In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **page-definition** element in the **gui-definition** element, and then to the **create-eo** element.

3 To specify a new object type for the objects you create, change the value of the **root-object** element. For example:

```
<root-object>Business</root-object>
```

Important: *This must be the name of the parent object.*

4 To specify a new name for the Create System Record tab, modify the value of the **tab-name** element.

```
<tab-name>Add Record</tab-name>
```

5 Save and close the file.

9.7.4 Configuring the History Page

The History page allows you to search for records, view a results list, and then view a history of changes to the record you select. You can configure the name of the tabbed heading for the page, the type of object returned, the appearance of the search and results pages, and the appearance of the merge history page. Do not modify the URL of the entrance to the History page. Following is a sample of the **history** element.

```
<history>
    <root-object>Company</root-object>
    <tab-name>History</tab-name>
    <tab-entrance>/stcedm/EnterXASearchAction.do</tab-entrance>
    <xa-search-page>
        <field-per-row>2</field-per-row>
    </xa-search-page>
    <search-result-list-page>
        <item-per-page>10</item-per-page>
        <max-result-size>200</max-result-size>
        <field-ref>Person.FirstName</field-ref>
        <field-ref>Person.LastName</field-ref>
        <field-ref>Phone.Phone</field-ref>
        <field-ref>Address.AddressLine1</field-ref>
    </search-result-list-page>
    <merge-history-key-field>
        <field-ref>Person.FirstName</field-ref>
        <field-ref>Person.LastName</field-ref>
    </merge-history-key-field>
</history>
```

To configure the History page

1 In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **page-definition** element in the **gui-definition** element, and then to the **history** element.

3 To specify a new object to be returned from a history search, change the value of the **root-object** element. For example:

```
<root-object>Customer</root-object>
```

Important: *This must be the name of the parent object.*

4 To specify a new name for the History tab, modify the value of the **tab-name** element.

```
<tab-name>Transaction History</tab-name>
```

5 To specify the number of fields to display in each row on the History search page, change the value of the **field-per-row** element in the **xa-search-page** element. For example:

```
<xa-search-page>
    <field-per-row>3</field-per-row>
</xa-search-page>
```

6 To specify the number of records to display on the History search results page, change the value of the **item-per-page** element in the **search-result-list-page** element. For example:

```
    <item-per-page>15</item-per-page>
```

7 To specify a maximum number of records to return from a History search, change the value of the **max-result-size** element in the **search-result-list-page** element. For example:

```
    <max-result-size>250</max-result-size>
```

8 To customize the fields that appear in the results list, do any of the following in the **search-result-list-page** element:

- Modify the value of an existing **field-ref** element to the simple field name of a field you want to appear in the results list.
- Add and name a new **field-ref** element (using the simple field name).
- Delete a **field-ref** element defining a field you do not want to appear in the results list.

9 To customize the fields that appear on the merge history page, do any of the following in the **merge-history-key-field** element:

- Modify the value of an existing **field-ref** element to the simple field name of a field you want to appear on the merge history page.
- Add and name a new **field-ref** element (using the simple field name).
- Delete a **field-ref** element defining a field you do not want to appear on the merge history page.

10 Save and close the file.

9.7.5 Configuring the Match Review Page

The Match Review page allows you to search for potential duplicate or assumed match records to compare, view a results list, and then view a comparison of the records you select. You can configure the name of the tabbed heading for the page, the type of object returned, and the appearance of the search and results pages. Do not modify the URL of the entrance to the Match Review page. Following is a sample of the **matching-review** element.

```
<matching-review>
    <root-object>Company</root-object>
    <tab-name>Matching Review</tab-name>
    <tab-entrance>/stcedm/EnterPDSearchAction.do</tab-entrance>
    <pd-search-page>
        <field-per-row>2</field-per-row>
    </pd-search-page>
```

```
        <search-result-list-page>
            <item-per-page>10</item-per-page>
            <max-result-size>250</max-result-size>
        </search-result-list-page>
    </matching-review>
```

To configure the Match Review page

1 In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.

2 Scroll to the **page-definition** element in the **gui-definition** element, and then to the **matching-review** element.

3 To specify a different object to be returned from the search, change the value of the **root-object** element. For example:

```
<root-object>Business</root-object>
```

Important: *This must be the name of the parent object.*

4 To specify a different name for the Matching Review tab, modify the value of the **tab-name** element.

```
<tab-name>Potential Duplicates</tab-name>
```

5 To specify the number of fields to display in each row on the Matching Review search page, change the value of the **field-per-row** element in the **pd-search-page** element. For example:

```
<pd-search-page>
    <field-per-row>3</field-per-row>
</pd-search-page>
```

6 To specify the number of records to display on the Matching Review search results page, change the value of the **item-per-page** element in the **search-result-list-page** element. For example:

```
    <item-per-page>15</item-per-page>
```

7 To specify a maximum number of results for a Matching Review search, change the value of the **max-result-size** element in the **search-result-list-page** element. For example:

```
    <max-result-size>100</max-result-size>
```

8 Save and close the file.

9.7.6 Configuring the Audit Log Pages

When enabled, the audit log stores a history of each instance in which information from the object tables in the master index database is accessed. The EDM allows you to search for and view the audit log entries. You can enable or disable the audit log in the Enterprise Data Manager file.

To configure the audit log pages

1 In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.

2. Scroll to the **page-definition** element in the **gui-definition** element, and then to the **audit-log** element.
3. To specify that records be written to the audit log, change the value of the **allow-insert** element to **true**. For example:

```
<audit-log>
    <allow-insert>true</allow-insert>
</audit-log>
```

4. To specify that records not be written to the audit log, change the value of the **allow-insert** element to **false**. For example:

```
<audit-log>
    <allow-insert>false</allow-insert>
</audit-log>
```

5. Save and close the file.

9.8 Configuring Implementation Information

Certain configuration information is defined automatically in the implementation details section of the Enterprise Data Manager file based on information you specify in the eView Wizard. Other information in this section must be configured. You can customize the implementation details by performing any of the following tasks.

- **Specifying the Master Controller JNDI Class** on page 161
- **Specifying Validation Services** on page 162
- **Setting Debug Options** on page 162
- **Configuring Security** on page 163

9.8.1 Specifying the Master Controller JNDI Class

The EDM must know the name of the Master Controller interface for the eView implementation. Only change this value if you created a custom Master Controller.

To specify the Master Controller JNDI class

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **impl-details** element, and then to the **master-controller-jndi-name** element.
3. Modify the value of the **master-controller-jndi-name** element to the name of the Master Controller JNDI class for your server.

```
<master-controller-jndi-name>ejb/CompanyMasterController
</master-controller-jndi-name>
```

4. Save and close the file.

9.8.2 Specifying Validation Services

Validation services verify processing codes for data entered into the master index database. Only change the names of the validation services if you created custom code list validators.

To specify the validation service

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **impl-details** element, and then to the **validation-service-jndi-name** element.
3. To change the name of the validator for processing codes stored in sbyn_common_header, modify the value of the **validation-service-jndi-name** element to the name of the JNDI class used for validation.

   ```
   <validation-service-jndi-name>ejb/CompanyCodeLookup
   </validation-service-jndi-name>
   ```
4. element.
5. To change the name of the validator for processing codes stored in sbyn_user_code, modify the value of the **usercode-jndi-name** element to the name of the JNDI class used for validation.

   ```
   <usercode-jndi-name>ejb/CompanyUserCodeLookup
   </usercode-jndi-name>
   ```
6. Save and close the file.

9.8.3 Setting Debug Options

When you first implement eView, you may want to view detailed debug information until you are certain the application is working as required. You can set debug options in the implementation details.

To set debug options

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **impl-details** element, and then to the **debug-flag** element.
3. Modify the value of the **debug-flag** element to indicate whether you want debug information logged. For example:

   ```
   <debug-flag>true</debug-flag>
   ```

   Specify **true** to log debug information; specify **false** to turn logging off.
4. Modify the value of the **debug-dest** element to indicate where to print debug information. For example:

   ```
   <debug-dest>console</debug-dest>
   ```

   Specify **console** to log debug information to a monitor; specify **file** to print to a file.
5. Save and close the file.

9.8.4 Configuring Security

You can specify whether or not authorization security for the EDM is enabled. Security is provided through the application or integration server.

Configuring Authorization Security

You can enable or disable authorization security for the EDM.

To enable authorization security

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **impl-details** element.
3. Change the value of the **enable-security** element to **true**. For example:

```
<enable-security>true</enable-security>
```

4. Save and close the file.

To disable authorization security

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **impl-details** element.
3. Change the value of the **enable-security** element to **false**. For example:

```
<enable-security>false</enable-security>
```

4. Save and close the file.

Specifying a new Field Masking Class

If you implement a custom class to define field masking logic, you must specify the new class in the implementation details. The custom class must be created as a custom plug-in in the eView Project. This class can be used to mask the value of any field for which the **is-sensitive** element is set to “true”.

To specify a new field masking class

1. In the Project Explorer pane of the Enterprise Designer, double-click **Enterprise Data Manager** under the **Configuration** folder in the Project you want to modify.
2. Scroll to the **impl-details** element.
3. Change the value of the **object-sensitive-plug-in-class** element to the name of the custom class. For example:

```
<object-sensitive-plug-in-class>com.stc.eindex.user.customfieldmasker
</object-sensitive-plug-in-class>
```

4. Save and close the file.

Appendix A

Field Notations

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

A.9 Defining Field Locations

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

A.9.1 ePath

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

ePath strings can be of four basic types:

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

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

Syntax

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

```
node{.node{‘[‘qualifier’]’}+}+.field
```

- **Nodes**
  The node specifies the node type, and optionally includes qualifiers to restrict the number of nodes. A node without any qualifier defaults to only the first node of the specified type. “node.*” is used to address a node rather than a field.
- **Qualifiers**
  Qualifiers restrict the number of nodes addressed at each level. The following qualifiers are allowed:
  - * (asterisk)
    Denotes all nodes of the specified type.
  - **int**
    Accesses the node by index.
  - **@keystring= valuestring**
    Accesses the node using a key-value pair. Only one instance of the node is addressed using keys. If a composite key is defined, then multiple key-value pairs can be separated by a comma 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 listed that refer to various elements of this object structure. A description of the data in the sample object structure referred by the ePath is also included.

```
Enterprise
    SystemObject - A 111
        Person
            FirstName
            LastName
            -Address
                AddressType = Home
                Street = 404 E. Huntington Dr.
                City = Monrovia
                State = CA
                PostalCode = 91016
            -Address
                AddressType = Office
                Street = 181 E. Huntington Dr.
                City = Monrovia
                State = CA
                PostalCode = 91016
            -Address
                AddressType = Billing
                Street = 100 Marine Parkway
                City = Redwood Shores
                State = CA
                PostalCode = 94065
```

- **Person.Address.City**
  Equivalent to **Person.Address[0].City**.
- **Person.FirstName** (uses Person as the context)
  Equivalent to **Enterprise.SystemObject[@SystemCode=A, @Lid= 111].Person.FirstName** with Enterprise as the context.
- **Person.Address[@AddressType=Home].City**
  Returns a single ObjectField reference to “Monrovia” (the City field of the home address).
- **Person.Address[City=Monrovia,State=CA].Street**
  Returns a list of ObjectField references: “404 E. Huntington 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: “404 E. Huntington Dr.”, “181 E. Huntington Dr.”, “100 Marine Parkway”. Note that all references to **Street** are returned.
- **Person.Address[2].***
  Addresses the second address object as an ObjectNode, instead of ObjectField.

A.9.2 Qualified Field Names

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

- **Fully qualified field names**—This type allows you to define fields within the context of the enterprise object; that is, the field name uses “Enterprise” as the root.

These are used in the **MatchingConfig** element of the Match Field file and to specify the fields in a query block in the Candidate Select file.

- **Qualified field names**—This type allows you to define fields within the context of the parent object; that is, the field name uses the name of the parent object as the root. These are used in the Candidate Select file to specify the source fields for the blocking query criteria.

Syntax

The syntax of a fully qualified field name is:

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

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

The syntax of a qualified field name is:

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

Example

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

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

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

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

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

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

A.9.3 Simple Field Names

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

Syntax

The syntax of a simple field name is:

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

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

Example

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

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

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

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

Glossary

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

assumed match
When the matching weight between two records is at or above a weight you specify, (depending on the configuration of matching parameters) the objects are an assumed match and are merged automatically (see "Automatic Merge").

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

Blocking Query
The query used during matching to search the database for possible matches to a new or updated record. This query makes multiple passes against the database using different combinations of criteria. The criteria is defined in the Candidate Select file.

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

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

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

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

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

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

duplicate threshold
The matching probability weight at or above which two records are considered to potentially represent the same entity.

EDM
See *Enterprise Data Manager*.

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

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

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

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

eView Manager Service
An eView component that provides an interface to all eView components and includes the primary functions of the master index. This component is configured by the Threshold file.

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

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

Field Validation file
The eView configuration file that specifies any custom Java classes that perform field validations when data is processed.

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

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

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

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

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

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

matching probability weight
An indicator of how closely two records match one another. The weight is generated using matching algorithm logic, and is used to determine whether two records represent the same object.

Matching Service
An eView component that defines the matching process. This component is configured by the Match Field file.

matching threshold
The lowest matching probability weight at which two records can be considered a match of one another.

matching weight *or* match weight
See *matching probability weight*.

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

merged profile
See *non-surviving profile*.

non-surviving profile
An object profile that is no longer active because it has been merged into another object profile. Also called a *merged profile*.

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

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

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

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

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

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

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

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

probability weight
See *matching probability weight*.

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

SBR
See *single best record*.

single best record
Also known as the SBR, this is the best representation of an entity's information. The SBR is populated with information from all source systems based on the survivor

strategies defined for each field. It is a part of an entity's enterprise object and is recalculated each time a system record is updated.

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

survivor calculator
The logic that determines which fields from which source systems should be used to populate the SBR. This logic is a combination of Java classes and user-configured logic contained in the Best Record file.

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

system
A computer application within your company where information is entered about the objects in the master index and that shares this information with the master index (such as a registration system). Also known as "source system" or "external system".

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

tab
A heading on an application window that, when clicked, displays a different type of information. For example, click the EDM tab on the Define Enterprise Object window to display the EDM attributes.

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

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

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

Index