SeeBeyond ICAN Suite

Implementing the SeeBeyond Match Engine with eView Studio

Release 5.0.1

SEEBEYOND®

Contents

List of Tables

Chapter 1

Introduction

This guide provides comprehensive information on working with the components of the SeeBeyond® Match Engine, and implementing the match engine with a SeeBeyond eView Studio (eView) 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, using a matching algorithm to identify data.

This guide includes complete descriptions of the components of the SeeBeyond Match Engine along with information about customizing each component. It also describes how to customize the eView configuration files to define standardization and matching fields for the SeeBeyond Match Engine. This guide is designed to be used in conjunction with the *eView Studio User's Guide* and the *eView Studio Configuration Guide*.

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 background information and instructions for implementing the SeeBeyond Match Engine with an eView master index. It includes descriptions of all components and the default configuration, and also provides instructions for customizing the eView configuration files for the match engine.

This guide does not include information or instructions for installing or configuring eView. These topics are covered in the appropriate user guide (for more information, see **"Supporting Documents" on page 12**).

1.1.1. Intended Audience

Any user who configures the eView Match Service (that is, the Match Field file), or who creates or customizes the match engine configuration files, should read this book. A thorough knowledge of eView is not needed to understand this guide, but familiarity with the eView configuration files (especially the Match Field file) is recommended. A general understanding of basic standardization and matching logic is helpful. It is presumed that the reader of this guide is familiar with the type of data being stored in the master index and the processing requirements for that data.

1.1.2. Document Organization

This guide is divided into eight chapters and two appendixes that cover the topics shown below.

- **Chapter 1** **"Introduction"** gives a general preview of this document—its purpose, scope, and organization—and provides sources of additional information.
- **Chapter 2** **"The SeeBeyond Match Engine"** gives an overview of the SeeBeyond Match Engine, the underlying matching algorithm, and the components of the match engine.
- **Chapter 3** **"Standardization Configuration Files"** gives an overview of the standardization configuration files and their purpose.
- **Chapter 4** **"Matching Configuration Files"** describes the matching configuration files and gives a reference of the comparison functions for matching.
- **Chapter 5** **"eView and the SeeBeyond Match Engine"** describes how the SeeBeyond Match Engine works with an eView master index.
- **Chapter 6** **"Person Data Type Configuration"**gives information and instructions for customizing the standardization and matching configuration files for person data, and gives instructions for customizing the Match Field file.
- **Chapter 7** **"Address Data Type Configuration"**gives information and instructions for customizing the standardization and matching configuration files for address fields, and gives instructions for customizing the Match Field file.
- **Chapter 8** **"Business Names Data Type Configuration"**gives information and instructions for customizing the standardization and matching configuration files for business name fields, and gives instructions for customizing the Match Field file.
- **Appendix A** **"Fine-tuning Weights and Thresholds"** describes the process of fine-tuning the matching logic and weight thresholds.
- **Appendix B** **"Match Configuration Comparison Functions"** lists and describes the comparison functions you can use with the SeeBeyond Match Engine.

1.2 Writing Conventions

Before you start using this guide, it is important to understand the conventions observed throughout this document.Table 1 lists these conventions.

Table 1 Special Notation Conventions

Text	Convention	Example
Titles of publications	Title caps in *italic* font	*eView Studio Configuration Guide*
Button, Icon, Command, Function, and Menu Names	**Bold** text	▪ Click **OK** to save and close. ▪ From the **File** menu, select **Exit**.

Table 1 Special Notation Conventions (Continued)

Text	Convention	Example
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 11**.
File Names and Paths	**Bold** text	The **matchConfigFile.cfg** file contains the matching rules for each field.
Notes	***Bold Italic*** text	***Note:*** *If a toolbar button is dimmed, you cannot use it with the selected component.*

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 implementing the SeeBeyond Match Engine.

- *eView Studio User's Guide*
- *eView Studio Configuration Guide*
- *eView Studio Reference Guide*

1.4 Online Documents

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

http://www.adobe.com

1.5 SeeBeyond Web Site

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

http://www.SeeBeyond.com

Chapter 2

The SeeBeyond Match Engine

The SeeBeyond Match Engine is the standard match engine for the master indexes created by eView. It is highly configurable in the eView environment and can be used to match on various types of data.

This chapter provides information about the configurable components of the match engine, and how the SeeBeyond Match Engine standardizes and matches data.

2.1 About the Matching Algorithm

The SeeBeyond Matching Engine compares records containing similar data types by calculating how closely the records match. The resulting comparison weight is either a positive or negative numeric value that represents the degree to which the two sets of data are similar. The match engine relies on probabilistic algorithms to compare data of a given type using a comparison function specific to the type of data being compared. The comparison functions for each matching field are defined in a match configuration file that you configure specifically for the type of data you are indexing. The formula used to determine the matching weight is based on either matching and unmatching probabilities or on specific agreement and disagreement weight ranges.

The SeeBeyond Match Engine is also designed to standardize freeform text fields, such as street address fields or business names. This allows the match engine to generate a more accurate weight for freeform data.

2.1.1. Standardization and Matching

The matching algorithm in the SeeBeyond Match Engine uses a proven methodology to process and weight records in the master index database. By providing both standardization and matching capabilities, the match engine allows you to condition data prior to matching. You can also use these capabilities to review legacy data prior to loading it into the database. This review helps you determine data anomalies, invalid or default values, and missing fields.

Both matching and standardization occur when two records are analyzed for matching probability. Before matching, certain fields are standardized and converted into their phonetic values if necessary. The match fields are analyzed and weighted according to the rules defined in a match configuration file. The weights for each field are combined to determine the overall matching weight for the two records. After the match engine has performed these steps, survivorship is determined by the master index, based on how the overall matching weight compares to the duplicate and match thresholds set for the master index. These thresholds are configured for the eView Manager Service in the Threshold file.

2.1.2. Data Types

You can standardize and match on different types of data with the SeeBeyond Match Engine. In its default implementation with eView, the match engine supports data standardization and matching on the three primary types of data listed below.

- Person Information
- Street Addresses
- Business Names

In addition, the SeeBeyond Match Engine provides comparison functions for matching on various types of fields contained within the primary data types, such as numbers, dates, Social Security Numbers, single characters, and so on.

When processing person information, the match engine assumes that each match field is stored in a separate field. For street address and business name processing, eView is configured to parse free-form fields for searching and matching. Each data type requires specific customizations to the Match Field file in the master index Project. The required customizations for each type are described in chapters 6 through 8 of this guide.

2.1.3. How it Works

The SeeBeyond Match Engine compares two records and returns a match weight indicating the likelihood of a match between the two records. There are three primary components of the match engine.

- **Configuration Files**
  The SeeBeyond Match Engines includes a set of files that define standardization and matching logic for all supported data types. You can customize these files to adapt the standardization and matching logic to your specific needs.
- **Standardization Engine**
  Standardization involves converting non-standard data into a standardized form so it is more accurate and efficient to process. Standardization consists of any one or more of the following actions:
  - **Parsing** — separating a free-text field into its individual components, such as street address information or a business name.
  - **Normalization** — changing the value of a field to a standard version, such as changing a nickname to a common name.
  - **Phonetic Encoding** — changing the value of a field to its phonetic version. The field to be converted can be the original field, a parsed field, a normalized field, or a parsed and normalized field.

  Using the person data type, for example, first names such as “Bill” and “Will” are normalized to “William”, which is then phonetically converted using Soundex.

Using the street address data type, street addresses are parsed into their component parts, such as house numbers, street names, and so on. The street name is then phonetically converted. Standardization logic is defined in the configuration files and in the **StandardizationConfig** section of the eView Match Field file, and is carried out prior to assigning match weights.

- **Match Engine**
  Matching involves comparing two standardized records and returning a weight that indicates the likelihood of a match between the two records. A higher weight indicates a greater likelihood of a match. Matching criteria and logic are defined in the match engine configuration files. The data fields that are sent to the SeeBeyond Match Engine for matching, known as the *match string*, are defined in the **MatchingConfig** module of the eView Match Field file. The match engine configuration files define how the match string is standardized and which matching rules to use to process each match field.

2.1.4. Matching Weight Formulation

The SeeBeyond Match Engine determines the matching weight between two records by comparing the match string fields using the rules defined in the match configuration file and by taking into account the matching logic specified for each field. The SeeBeyond Match Engine can use either matching (m) and unmatching (u) conditional probabilities or agreement and disagreement weight ranges to fine-tune the match process. It uses the underlying algorithm to arrive at a match weight for each match string field. A weight is generated for each field in the match string indicating the level of match between each field. The weights assigned to each field are then summed together for a total, composite matching weight. Agreement and disagreement weight ranges or m-probabilities and u-probabilities are defined in the match configuration file.

Matching and Unmatching Probabilities

When matching and unmatching conditional probabilities are used, the match engine uses a logarithmic formula to determine agreement and disagreement weights between fields. The m-probabilities and u-probabilities you specifiy determine the maximum agreement weight and minimum disagreement weight for each field, and so define the agreement and disagreement weight ranges for each field and for a complete record. These probabilities allow you to specify which fields provide the most reliable matching information and which provide the least. For example, in person matching, the gender field is not as reliable as the SSN field for determining a match since a person's SSN is more specific. Therefore, the SSN field should have a higher m-probability than the gender field. The more reliable the field, the greater the m-probability for that field should be.

If a field matches between two records, an agreement weight, determined by the logarithmic formula using the m-probability and u-probability, is added to the composite match weight for the record. If the fields disagree, a disagreement weight is subtracted from the composite match weight. m-probabilities and u-probabilities are expressed as decimals between one and zero (excluding one and zero).

Agreement and Disagreement Weight Ranges

Defining agreement and disagreement weight ranges is a more direct way to implement m-probabilities and u-probabilities. Like probabilities, the maximum agreement and minimum disagreement weights you define for each field allow you to define the relative reliability of each field; however, the match weight has a more linear relationship with the numbers you specify. When you use agreement and disagreement weight ranges to determine the match weight, you define a maximum weight for each field when they are in complete agreement and a minimum weight for when they are in complete disagreement. The SeeBeyond Match Engine assigns a matching weight to each field that falls between the agreement and disagreement weights specified for the field. This provides a more convenient and intuitive representation of conditional probabilities.

Using the SSN and gender field example above, the SSN field would be assigned a higher maximum agreement weight and a lower minimum disagreement weight than the gender field because it is more reliable. If you assign a maximum agreement weight of “10” and two SSNs match, the match weight for that field is “10”. If you assign a minimum disagreement weight of “-10” and two SSNs are in complete disagreement, the match weight for that field is “-10”.

Chapter 3

Standardization Configuration Files

The standardization configuration files for the SeeBeyond Match Engine must follow certain rules for formatting and interdependencies. This chapter provides an overview of the types of configuration files provided for standardization, the architecture of those files, and formatting descriptions.

3.1 About Standardization Configuration Files

The standardization configuration files define additional logic used by the SeeBeyond Match Engine to standardize specific data types. This logic helps define how fields in incoming records are parsed, standardized, and classified for processing. Standardization files include data patterns files, category files, clues value tables, type key tables, and constants files. Reference files define special characters that are stripped from the initial data string to aid in standardization or that are used to join different parts of an address or business name.

The standardization configuration files are stored in the eView Project, and appear as nodes in the **Standardization Engine** node of the Project.

3.1.1. Standardization Configuration File Types

Several different types of configuration files are included with the SeeBeyond Match Engine, each providing specific information to help the engine standardize and match data according to requirements.

- **Category Files**—The SeeBeyond Match Engine uses category files when processing person or business names. These files list common values for certain types of data, such as titles, suffixes, and nicknames for person names or industries and organizations for business names. Category files also define standardized versions of each term or classify the terms into different categories, and some files perform both functions. When processing address files, category files named “clues files” are used.
- **Clues Files**—The SeeBeyond Match Engine uses clues files when processing address data types. These files list general terms used in street address fields, define standardized versions of each term, and classify the terms into various component types using predefined address tokens. These files are used by the standardization engine to determine how to parse a street address into its various components.

Clues files provide clues in the form of tokens to help the engine recognize the component type of certain values in the input fields.

- **Constants Files**—The SeeBeyond Match Engine refers to constants files for information about the standardization files, such as the maximum length of the files. For the address data type, the constants file also describes input and output field lengths.
- **Patterns Files**
  The patterns files specify how incoming data should be interpreted for standardization based on the format, or pattern, of the data. These files are used only for processing data contained in freeform text fields that must be parsed prior to matching (such as street address fields or business names). Patterns files list possible input data patterns, which are encoded in the form of tokens. Each token signifies a specific component of the freeform text field. For example, in a street address field, the house number is identified by one token, the street name by another, and so on. Patterns files also define the format of the output fields for each input pattern.
- **Key Type Files**
  For business name processing, the SeeBeyond Match Engine refers to a number of key type files for processing information. These files generally define standard versions of terms commonly found in business names and some classify these terms into various components or industries. These files are used by the standardization engine to determine how to parse a business name into its different components and to recognize the component type of certain values in the input fields.
- **Reference Files**
  Reference files define general terms that appear in input fields for each data type. Some reference files define terms to ignore, and some define terms that indicate the business name is continuing. For example, in business name processing “and” is defined as a joining term. This helps the standardization engine to recognize that the primary business name in “Martin and Sons, Inc.” is “Martin and Sons” instead of just “Martin”. Reference files can also define characters to be ignored by the standardization engine.

Chapter 4

Matching Configuration Files

The matching configuration files for the SeeBeyond Match Engine must follow certain rules for formatting and interdependencies. This chapter provides an overview of the two matching configuration files provided, the architecture of those files, and formatting descriptions. It also provides a reference of comparison functions for use in the match configuration file.

4.1 About Matching Configuration Files

The matching configuration files define how the SeeBeyond Match Engine processes records to assign matching probability weights, allowing the master index to identify matches, potential duplicates, and non-matches. These files consist of two configurable files, the match configuration file and the match constants file. Together these files define additional logic for the SeeBeyond Match Engine to use when determining the matching probability between two records. A third file, the internal match constants file, is read-only and used internally by the match engine. It defines each comparison function and the comparison options.

The matching configuration files are very flexible, allowing you to customize the matching logic according to the type of data stored in the master index and for the record matching requirements of your business. The matching configuration files are stored in the eView Project, and appear as nodes in the **Match Engine** node of the Project. The SeeBeyond Engine standardizes the data prior to matching, so the match process is performed against the standardized data.

4.2 The Match Configuration File

The match configuration file, **matchConfigFile.cfg**, contains the matching logic for each field on which matching is performed. This file handles the matching logic for the three primary data types (person names, business names, and addresses), and can also handle generic data types, such as dates, numbers, social security numbers, and characters.

The match configuration file defines matching logic for each field on which matching is performed. The SeeBeyond Match Engine provides several comparison functions that you can call in this file to fine-tune the match process. Comparison functions contain the logic to compare different types of fields between records in very specific ways in order to arrive at a match weight for each field. These functions allow you to define how matching is performed for different data types, and can be used in conjunction with either matching and unmatching probabilities or agreement and disagreement weight ranges for each field. This file also defines how to handle missing fields.

4.2.1. Match Configuration File Format

The match configuration file is divided into two sections. The first section consists of one line that indicates the matching probability type. The second section consists of the matching rules to use for each match field.

Sample

Following is an excerpt from the default match configuration file. This excerpt illustrates the components that are described in the following sections.

```
ProbabilityType           1

FirstName               15  0   uf    0.99  0.001   15  -5
LastName                15  0   ul    0.99  0.001   15  -5
String                  25  0   ua    0.99  0.001   10  -5
DateDays                20  0   dD    0.99  0.001   10  -10 y 15 30
DateMonths              20  0   dM    0.99  0.001   10  -10 n
DateHours               20  0   dH    0.99  0.001   10  -10 y 30 60
DateMinutes             20  0   dm    0.99  0.001   10  -10 y 300 600
DateSeconds             20  0   ds    0.99  0.001   10  -10 y 75 60
Numeric                 15  0   n     0.99  0.001   10  -10 y 8
Integer                 15  0   nI    0.99  0.001   10  -10 n
Real                    15  0   nR    0.99  0.001   10  -10 n
Char                    1   0   c     0.99  0.001   5   -5
pro                     15  0   p     0.99  0.001   10  -10 20 5 5
```

Probability Type

The first line of the match configuration file defines the probability type to use for matching. Specify “0” (zero) to use m-probabilities and u-probabilities to determine a field’s match weight; specify “1” (one) to use agreement and disagreement weight ranges to determine a field’s match weight. If the probability type is set to use agreement and disagreement weight ranges, the **m-prob** and **u-prob** columns in the matching rules section are ignored. If the probability type is set to use m-probabilities and u-probabilities, the **agreement-weight** and **disagreement-weight** columns in the matching rules section are ignored. The default is to use agreement and disagreement weight ranges because they are more intuitive.

Matching Rules

The section after the first line of the match configuration file contains match field rows, which define how each field will be matched. The syntax for this section is:

```
match-type maximum-length null-field function m-prob u-prob
agreement-weight disagreement-weight parameters
```

Table 2 describes each element in a match field row.

Table 2 Match Configuration File Columns

Column Number	Column Name	Description
1	match-type	A value that indicates to the SeeBeyond Match Engine how each field should be weighted. Each field specified for matching in the **MatchingConfig** section of the Match Field file must have a match type corresponding to a value in this column.
2	size	The number of characters in the field on which matching is performed, beginning with the first character. For example, to match on only the first four characters in a 10-digit field, the value of this column should be "4".
3	null-field	An index that specifies how to calculate the total weight for null fields or fields that only contain spaces. You can specify any of the following values: ▪ **0** — (zero) If one or both fields are empty, the weight used for the field is 0 (zero). ▪ **1** — (one) If both fields are empty, the agreement weight is used; if only one field is empty, the disagreement weight is used. ▪ **a#** — An "a" followed by a number specifies to use the agreement weight if both fields are empty. The agreement weight is divided by the number following the "a" to obtain the match weight for that field. If no number is specified, the default is "2". You can specify any number from 1 through 10. ▪ **d#** — A "d" followed by a number specifies to use the disagreement weight if only one field is empty. The disagreement weight is divided by the number following the "d" to obtain the match weight for the field. If no number is specified, the default is "2". You can specify any number from 1 through 10. ***Note:*** *In the above descriptions, the agreement and disagreement weights are either specified in this file or calculated using a logarithmic formula based on the m and u-probabilities (depending on the probability type).*
4	function	The type of comparison to perform when weighting the field. For information about the available comparison functions, see **Appendix B**.
5	m-prob	The initial probability that the specified field in two records will match if the records match. The probability is on a scale from 0 to 1.

Table 2 Match Configuration File Columns

Column Number	Column Name	Description
6	u-prob	The initial probability that the specified field in two records will match if the records do not match. The probability is on a scale from 0 to 1.
7	agreement-weight	The matching weight to be assigned to a field given that the fields match between two records. This number can be between 0 and 100 and represents the maximum match weight for a field.
8	disagreement-weight	The matching weight to be assigned to a field given that the fields do not match between two records. This number can be between 0 and -100 and represents the minimum match weight for a field.
9	parameters	The parameters correspond to the comparison function. Some comparison functions do not take any parameters, and some take multiple parameters. For additional information about parameters, see **Appendix B**.

4.2.2. Matching Comparison Functions

Match field comparison functions compare the values of a field in two records to determine whether the fields match. The fields are then assigned a matching weight based on the results of the comparison function. You can use several different types of comparison functions in the match configuration file to define how the SeeBeyond Match Engine should match two records. The SeeBeyond Match Engine also provides several options to use with each function. Table 3 summarizes each comparison function. A complete reference of the comparison functions and options is included in **Appendix B**.

Table 3 Comparison Functions

Comparison Function	Description
b1	Based on the *Bigram* algorithm, this function compares two strings using all combinations of two consecutive characters and returns the total number of combinations that are the same.
b2	Similar to the standard Bigram comparison function (b1), but allows for character transpositions.
u	Based on the *Jaro* algorithm, this function compares two strings taking into account uncertainty factors, such as string length, transpositions, and characters in common.

Table 3 Comparison Functions

Comparison Function	Description
ua	Based on the *Jaro* algorithm with variants of Winkler/Lynch and McLaughlin , this function is similar to the basic uncertainty comparison function (u), but increases the agreement weight if the initial characters of each string are exact matches. This comparison function takes into account key punch and visual memory errors.
uf	Based on the simplex uncertainty comparison function (u), this function is designed to specifically weight first name values. The string is analyzed and the weight adjusted based on statistical data.
ul	Based on the simplex uncertainty comparison function (u), this function is designed to specifically weight last name values. The string is analyzed and the weight adjusted based on statistical data.
un	Based on the simplex uncertainty comparison function (u), this function is designed to specifically weight house number values. The string is analyzed and the weight adjusted based on statistical data.
us	A custom uncertainty comparison function that compares two strings taking into account such uncertainty factors as string length, transpositions, key punch errors, and visual memory errors. Unlike the uncertainty comparison function ("u"), this function handles diacritical marks. This function also improves processing speed.
c	Compares string fields character by character. Each character must match in order for an agreement weight to be assigned.
n	Compares numeric fields using a relative distance value to determine the match weight. As the difference between the two fields increases, the match weight decreases. Once the difference is beyond the relative distance, a disagreement weight it assigned.
nI	Compares integer fields using a relative distance comparison. This comparison function is based on the standard numeric comparison function (n).
nR	Compares fields containing real numbers using a relative distance comparison. This comparison function is based on the standard numeric comparison function (n).
nS	Compares social security numbers, taking into account any of these options: ▪ Field length ▪ Character types ▪ Invalid values

Table 3 Comparison Functions

Comparison Function	Description
dY	Compares year values using relative distance values prior to and following the given year to determine the match weight. As the difference between the two fields increases, the match weight decreases. Once the difference is beyond the relative distance, a disagreement weight it assigned. The date comparison functions handle Gregorian years.
dM	Compares the month and year using a relative distance, as described above for the year comparison function (dY).
dD	Compares the day, month, and year using a relative distance, as described above for the year comparison function (dY).
dH	Compares the hour, day, month, and year using a relative distance, as described above for the year comparison function (dY).
dm	Compares the minute, hour, day, month, and year using a relative distance, as described above for the year comparison function (dY).
ds	Compares the second, minute, hour, day, month, and year using a relative distance, as described above for the year comparison function (dY).
p	Prorates the disagreement weight for a date or numeric field based on values you specify. Differences greater than the amount you specify receive the full disagreement weight.

4.3 The Match Constants File

The match constants file, **matchConstants.txt**, defines certain configurable constants used by the match engine. This file includes four parameters, but currently only the first parameter, **nFields**, is used. This parameter defines the maximum number of fields being used for matching. This must be equal to or greater than the number of fields defined in the **match-columns** element of the eView Match Field file. The match constants file defines the following constants for the match engine.

nFields

This constant defines the maximum number of different matching fields. You can enter any integer, but this number must be equal to or greater than the number of fields defined in the match-columns elements in the eView Match Field file.

maxFreqTableSize

This constant is only used when frequency tables are used. This is not currently available, and this constant is ignored.

maxNumberTables

This constant is only used when frequency tables are used. This is not currently available, and this constant is ignored.

mcls

This constant is only used when the generic-type frequency tables are used. This is not currently available, and this constant is ignored.

Chapter 5

eView and the SeeBeyond Match Engine

Implementing the SeeBeyond Match Engine in an eView master index requires certain configurations to the Match Field file in the eView Project. You can also customize the SeeBeyond Match Engine configuration files to better suit your data standardization and matching requirements.

This chapter provides information about the required configurations, and how the Match Field file corresponds to the configuration files.

5.1 SeeBeyond Match Engine and the Master Index

The eView master index uses the SeeBeyond Match Engine specifically for standardization and probabilistic weighting, while the master index determines survivorship. This process relies on the logic specified in the configuration files of the eView Project and of the SeeBeyond Match Engine.

5.1.1. Searching and Matching in the Master Index

When a new record is passed to the master index database, the index selects a subset of possible matches from the database. The index then uses the SeeBeyond Match Engine's matching algorithm to assign a matching probability weight for each record in this subset (known as the *candidate selection pool*).   To create the candidate selection pool, the index makes series of query passes of the existing data, searching for matches on specific combinations of data. These combinations are defined by the blocking query that is defined in the Candidate Select file and specified in the Threshold file.

Matching is performed on the data elements included in the match string in the Match Field file. Each data element is assigned a matching weight. The weights for each data element are summed to determine the matching probability weight for the entire record. Before matching on certain fields, such as the first name, the index may standardize that field based on information in certain standardization files. You can customize how each data element is weighted in the match configuration file.

5.1.2. The Standardization and Matching Process

The standardization and matching processes use logic that is defined by a combination of SeeBeyond Match Engine configuration files and eView configuration files. During the standardization and match processes, the following occurs.

1. The SeeBeyond Match Engine receives the match string from an incoming record, as defined by the **match-columns** elements of the **MatchingConfig** section of the Match Field file.
2. The SeeBeyond Match Engine standardizes the fields specified for parsing, normalization, and phonetic encoding. These fields are defined in the **StandardizationConfig** section of the Match Field file, and the rules for standardization are defined in the SeeBeyond Match Engine standardization configuration files.
3. eView queries the database for records that are possible matches using the specified blocking query. If the blocking query uses standardized or phonetic fields, the criteria values are obtained from the database.
4. For each possible match, eView creates a match string (based on the match columns in **MatchingConfig**) and sends the string to the SeeBeyond Match Engine.
5. The SeeBeyond Match Engine matches the incoming record against each possible match, producing a matching weight. Matching is performed using the weighting rules defined in the match configuration file.

5.1.3. The Match String

The data string that is passed to the SeeBeyond Match Engine for match processing is called the *match string*, and is defined in the **MatchingConfig** section of the Match Field file. The SeeBeyond Match Engine configuration files, the blocking query, and the matching configuration are closely linked in the search and matching processes. The blocking query defines the select statements for creating the candidate selection pool during the matching process. The matching configuration defines the match string that is passed to the SeeBeyond Match Engine. Finally, the SeeBeyond Match Engine configuration files define how the match string is processed.

The SeeBeyond Match Engine configuration files are dependent upon the match string, and it is very important when you modify the match string to ensure that the match type you specify corresponds to the correct row in the match configuration file (**matchConfigFile.cfg**). For example, if you are using person matching and add "MaritalStatus" as a match field, you need to specify a match type for the MaritalStatus field that is listed in the first column of the match configuration file. You must also make sure that the matching logic defined in the corresponding row of the match configuration file is defined appropriately for matching on the MaritalStatus field.

5.1.4. Field Identifiers

The SeeBeyond Match Engine uses field identifiers (field IDs) to determine how to process fields that are defined for normalization or parsing. The IDs are defined internally in the match engine, and are referenced in the Match Field file. The field IDs for business names specify which business type key file to use for standardization.

You can only specify the predefined field IDs that are listed in Table 4.

Table 4 Standardization Field Identifiers

Use this field ID ...	for these fields ...
Person Name Standardization Field Identifiers	
FirstName	Any first name field.
MiddleName	Any middle name field.
LastName	Any last name field.
Address Standardization Field Identifiers	
HouseNumber	The parsed house number field, which stores the house, rural route, and PO box numbers.
RuralRouteIdentif	The parsed house number field, which stores the house, rural route, and PO box numbers.
BoxIdentif	The parsed house number field, which stores the house, rural route, and PO box numbers.
OrigStreetName	The parsed street name field, which stores the street name, rural route descriptor, or PO box type.
RuralRouteDescript	The parsed street name field, which stores the street name, rural route descriptor, or PO box type.
BoxDescript	The parsed street name field, which stores the street name, rural route descriptor, or PO box type.
PropDesPrefDirection	The parsed address field containing the standardized street direction. This field ID handles cases where the direction is a prefix to the property description.
PropDesSufDirection	The parsed address field containing the standardized street direction. This field ID handles cases where the direction is a suffix to the property description.
StreetNamePrefDirection	The parsed address field containing the standardized street direction. This field ID handles cases where the direction is a prefix to the street name.
StreetNameSufDirection	The parsed address field containing the standardized street direction. This field ID handles cases where the direction is a suffix to the street name.
StreetNameSufType	The parsed address field containing the standardized street type. This field ID handles cases where the street type is a suffix to the street name.

Table 4 Standardization Field Identifiers

Use this field ID ...	for these fields ...
StreetNamePrefType	The parsed address field containing the standardized street type. This field ID handles cases where the street type is a prefix to the street name.
PropDesSufType	The parsed address field containing the standardized street type. This field ID handles cases where the street type is a suffix to the property description.
PropDesPrefType	The parsed address field containing the standardized street type. This field ID handles cases where the street type is a prefix to the property description.
Business Name Standardization Field Identifiers	
PrimaryName	The field containing the parsed name in a freeform text business name field.
OrgTypeKeyword	The field containing the parsed organization type in a freeform text business name field.
AssocTypeKeyword	The field containing the parsed association type in a freeform text business name field.
IndustrySectorList	The field containing the parsed industry sector in a freeform text business name field.
IndustryTypeKeyword	The field containing the parsed industry type in a freeform text business name field.
AliasList	The field containing the parsed alias in a freeform text business name field.
Url	The field containing the parsed URL in a freeform text business name field.

5.1.5. Match and Standardization Types

The Match Field file in the eView Project uses indicators to reference the type of matching and standardization to perform on each field. You must specify one of these indicators, called *match types* and *standardization types,* for the fields you define for standardization or matching. The match types correspond to the match types listed in the first column of the match configuration file (**matchConfigFile.cfg**). The standardization types are defined internally in the match engine. The types let the SeeBeyond Match Engine know how to process each field.

Table 5 lists the default standardization types; Table 6 lists the default match types. You can modify the match type names, but not the standardization type names. For more information about match and standardization types, see "Match and Standardization Types" in Appendix B of the *eView Studio User's Guide*. Note that the match types you can specify in the Match Field file (listed in Table 6) are not the same values you specify for the **Match Type** field drop-down list in the eView Wizard.

Table 5 Standardization Types

This indicator ...	processes this data type ...
Address	Free-form street address fields.
PersonName	Pre-parsed name fields (including any first, middle, last, or alias names).
BusinessName	Free-form business names.

The standardization types listed above correspond to the match types listed below. You can also specify miscellaneous match types, which do not correspond to any standardization types.

Table 6 Match Types

This indicator ...	processes this type of data
Business Name Match Types	
PrimaryName	The parsed name field of a business name.
OrgTypeKeyword	The parsed organization type field of a business name.
AssocTypeKeyword	The parsed association type field of a business name.
AliasList	The parsed alias type field of a business name.
IndustrySectorList	The parsed location type field of a business name.
IndustryTypeKeyword	The parsed industry type field of a business name.
Url	The parsed URL field of a business name.
Address Match Types	
StreetName	The parsed street name field of a street address.
HouseNumber	The parsed house number field of a street address.
StreetDir	The parsed street direction field of a street address.
StreetType	The parsed street type field of a street address.
Person Name Match Types	
FirstName	A first name field, including middle name, alias first name, and alias middle name fields.
LastName	A last name field, including alias last name fields.
Date Match Types	

Table 6 Match Types

This indicator ...	processes this type of data
DateDays	The day, month, and year of a date field.
DateMonths	The month and year of a date field.
DateHours	The hour, day, month, and year of a date field.
DateMinutes	The minute, hour, day, month, and year of a date field.
DateSeconds	The seconds, minute, hour, day, month, and year of a date field.
Miscellaneous Match Types	
String	A generic string field.
Numeric	A numeric field.
Integer	A field containing integers.
Real	A field containing real numbers.
SSN	A field containing a social security number.
Char	A field containing a single character.
pro	Any field on which you want the SeeBeyond Match Engine to use prorated weights.
Exac	Any field you want the SeeBeyond Match Engine to match character for character.

5.1.6. Customizing SeeBeyond Match Engine Configuration Files

The SeeBeyond Match Engine configuration files are designed to perform very specific functions in the standardization and match processes. These files should only be modified by personnel with an understanding of the SeeBeyond Match Engine and an understanding of the data integrity requirements of your organization. SeeBeyond recommends that any modifications to both the eView configuration files and the SeeBeyond Match Engine configuration files be made while the master index is in the pre-production stages. Modifying the files after the master index has moved into production may cause variances in matching weights and data processing.

The most common modifications to the SeeBeyond Match Engine configuration files are generally in the match configuration file, where you can fine-tune the weighting process. This file defines probabilities used by the algorithm to determine a matching probability weight for each match field. You can use the match comparison functions provided by the SeeBeyond Match Engine to fine-tune the matching logic in this file. Another common modification is inserting additional names or terms into category files, such as the first name category file (**personFirstNameUS.dat**).

Depending on your data requirements, you might need to modify additional standardization files. Some of the patterns files (most notably the address patterns file) are very complex, and should only be modified by personnel who thoroughly understand the defined patterns and tokens.

5.2 Matching Service Configuration

To configure an eView master index for specific data types and the SeeBeyond Match Engine, you must customize the Matching Service by modifying the Match Field file in the eView Project. The Match Field file has three sections to customize: standardization configuration, match field configuration, and match and standardization engine configuration.

5.2.1. Standardization Configuration

The **StandardizationConfig** section of the Match Field file determines which fields are normalized, parsed, or phonetically encoded. This section of the file includes the following structures.

- **Normalization Structures** on page 32
- **Standardization Structures (Parsing and Normalization)** on page 33
- **Phonetic Encoding Structures** on page 34

This section defines fields that will be normalized, fields that will be parsed and normalized, and fields that will be phonetically encoded. The standardization types you specify in this section correspond to the match configuration file; the field IDs you can specify are listed in **Table 4 on page 28**.

Normalization Structures

The normalization structure defines fields that are already parsed, but need to be normalized. It also tells the SeeBeyond Match Engine where to place the normalized data in the object structure. Matching on any of these fields is determined by the match string and the logic is defined in the match configuration file.

Of the three data types processed by the SeeBeyond Match Engine, it only expects the person name data type to provide information in parsed fields; that is, the first, last, and middle names appear in separate fields, as do the suffix, title, and so on. The person standardization files define logic for normalizing person name fields. By default, only person first, last, and middle names and the alias first, last, and middle names are defined for normalization. You can specify additional name fields for normalization, such as maiden name, spouse’s name, and so on.

Defining new Fields for Normalization

The fields you define for normalization in the Match Field file can include any name fields. If you define normalization for fields that are not currently defined for normalization in the Match Field file, you must make the following modifications to the remaining configuration files.

1. Define the normalization structure, using the appropriate standardization type (PersonName) and field IDs (FirstName, MiddeName, or LastName).
2. Add the new fields that will store the normalized version of the original field value to the appropriate objects in the eView Object Definition file.

3 If any the normalized fields are to be used for blocking, modify the Candidate Select file by adding the new fields to the blocking query being used.

4 Regenerate the eView application in Enterprise Designer to include the new fields in the database creation script, the outbound Object Type Definition (OTD), and the method OTD.

Defining new Normalized Fields for Matching

If you want to match on the new fields storing the normalized data, you must perform the following steps.

1 Determine the match type or the match comparison function you want to use to match the normalized data, and modify the match configuration file (**matchConfigFile.cfg**) if needed.

2 Add the new normalized field to the **match-columns** element of the **MatchingConfig** section of the Match Field file, making sure to use the appropriate match type from the match configuration file.

Standardization Structures (Parsing and Normalization)

The fields that must be parsed, and possibly normalized, are defined in a standardization structure in **StandardizationConfig**. The standardization structure tells the SeeBeyond Match Engine where to place the standardized information extracted from the fields that are parsed. The target fields you specify for standardization facilitate searching by the parsed fields. Matching on any of these fields is determined by the match string and the logic is defined in the match configuration file.

The SeeBeyond Match Engine expects business names and street address information in freeform text fields that must be parsed and normalized prior to matching. The logic for parsing and normalizing street address information is contained in the address standardization files; the logic for parsing and normalizing business names is contained in the business standardization files. You can customize the standardization of these data types by modifying the appropriate patterns file.

Defining new Fields for Standardization

The fields you define for standardization in the Match Field file can include any street address field or fields containing business names. Perform the following tasks if you need to define one of these field types for standardization.

1 If necessary, modify the patterns file (you can define new input and output patterns or modify existing ones).

2 Define the standardization structure, using the appropriate standardization type (BusinessName or Address) and field IDs (described in **Table 4 on page 28**).

3 Add the new fields that will store the parsed or normalized data to the appropriate objects in the Object Definition file.

4 If any the parsed or normalized fields are to be used for blocking, modify the Candidate Select file by adding the new fields to the blocking query being used.

5 Regenerate the eView application in Enterprise Designer to include the new fields in the database creation script, the outbound Object Type Definition (OTD), and the method OTD.

Defining new Standardized Fields for Matching

If you want to match on the new fields storing standardized data, you must perform the following steps.

1 Determine the match type or the match comparison function you want to use to match the parsed data, and modify the match configuration file (**matchConfigFile.cfg**) if needed.

2 Add the new standardized field to the **match-columns** element of the **MatchingConfig** section of the Match Field file, making sure to use the appropriate match type from the match configuration file.

Phonetic Encoding Structures

The fields that must be phonetically encoded are defined in a phonetic encoding structure in **StandardizationConfig**. The phonetic encoding structure tells the SeeBeyond Match Engine where to place the phonetic data created from the fields that are encoded. You can define any field in the object structure for phonetic encoding.

Defining new Fields for Phonetic Encoding

The fields you define for phonetic encoding in the Match Field file can include any field.

1 Determine the type of phonetic encoder to use to convert the field. You can use either Soundex or NYSIIS encoding.

2 Define the phonetic encoding structure, as described in chapters 6 through 8.

3 Add the new fields that will store the phonetic version of the original field value to the appropriate objects in the Object Definition file.

4 If any the phonetic fields are to be used for blocking, modify the Candidate Select file by adding the new fields to the blocking query being used.

5 Regenerate the eView application in Enterprise Designer to include the new fields in the database creation script, the outbound OTD, and the method OTD.

5.2.2. Matching Configuration

The **MatchingConfig** section determines which fields are passed to the SeeBeyond Match Engine for matching (the match string). If you are matching on fields parsed from a freeform text field, define each individual parsed field you want to use for matching. The default fields listed in **MatchingConfig** depend on the fields you specified for matching in the eView Wizard.

The match types you can use for each field in this section are defined in the first column of the match configuration file. Make sure the match type you specify has the correct matching logic defined in the match configuration file.

5.2.3. Configuring the Match and Standardization Engines

To configure the master index to use the SeeBeyond Match Engine for standardization and matching, you must specify the SeeBeyond Match Engine in the **MEFAConfig** section of the Match Field file. This includes modifying the **standardizer-api**, **standardizer-config**, **matcher-api**, and **matcher-config** elements in **MEFAConfig**. For more information, see "MEFA Configuration" in Chapter 6 of the *eView Studio Configuration Guide*.

Following is a sample illustrating the four elements that define the match and standardization engines.

```
<MEFAConfig module-name="MEFA"
parser-class="com.stc.eindex.configurator.impl.MEFAConfiguration">
...
     <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>
</MEFAConfig>
```

To Specify the Standardization Engine

To specify the SeeBeyond Match Engine as the standardization engine, you must specify the API and configuration classes for the engine.

1. Open the Match Field file in the eView Project.
2. Scroll to the **MEFAConfig** section of the file.
3. In the **standardizer-api** element, enter **com.stc.eindex.matching.adapter.SbmeStandardizerAdapter**.
4. In the **standardizer-config** element, enter **com.stc.eindex.matching.adapter.SbmeStandardizerAdapter Config**.
5. Save and close the file.

To Specify the Match Engine

To specify the SeeBeyond Match Engine as the match engine, you must specify the API and configuration classes for the engine.

1. Open the Match Field file in the eView Project.
2. Scroll to the **MEFAConfig** section of the file.
3. In the **matcher-api** element, enter **com.stc.eindex.matching.adapter.SbmeMatcherAdapter**.
4. In the **matcher-config** element, enter **com.stc.eindex.matching.adapter.SbmeMatcherAdapter Config**.
5. Save and close the file.

Chapter 6

Person Data Type Configuration

Processing person data involves normalizing and phonetically encoding certain fields prior to matching. This chapter describes the configuration files that define person processing logic, and provides instructions for modifying the Match Field file for processing person data.

6.1 Person Matching Overview

Matching on the person data type includes standardizing and matching on demographic information about a person. The SeeBeyond Match Engine can create normalized and phonetic values for person data. Several configuration files designed specifically to handle person data are included to provide additional logic for the standardization and phonetic encoding process. These include files referenced by the standardization engine for information about how to standardize different types of person data. The SeeBeyond Match Engine can phonetically encode any field you specify, with some modification to the standardization files. It can also match on any field, as long as the match type for the field is defined in the match specifications file (**matchConfigFile.cfg**).

In addition, when storing person information, you might want to standardize addresses to enable searching against address information. This requires working with the address configuration files, described in **Chapter 7**, **“Address Data Type Configuration”**.

6.1.1. Person Data Processing Fields

When matching on person data, not all fields in a record need to be processed by the SeeBeyond Match Engine. The match engine only needs to process fields that must be parsed, normalized, or phonetically converted, and the fields against which matching is performed. These fields are defined in the Match Field file, and processing logic for each field is defined in the standardization and matching configuration files.

Match String Fields

The match string processed by the SeeBeyond Match Engine is defined by the match fields specified in the Match Field file. The match engine can process any combination of fields you specify for matching. By default, the match configuration file includes rows specifically for matching on first name, last name, social security number, and dates (such as a date of birth). It also includes a row for matching a single character, such as might be the case in a gender field. You can use any of the existing rows for matching, or you can create new rows for the fields you want to match. Any fields for which you specify a Match Type in the eView Wizard are added to the match string.

Standardized Fields

The SeeBeyond Match Engine expects person data to be provided in separate fields within a single record, meaning that no parsing is required of the name fields prior to normalization. The match engine is designed to normalize only first and last name fields in person data. Finally, the match engine can phonetically convert any field you choose using either the NYSIIS or Soundex phonetic encoder.

The Object Structure

The fields you specify for person name matching in the eView Wizard are automatically defined for standardization and phonetic encoding. If you specify the appropriate match types in the eView Wizard, the following fields are automatically added to the object structure and database creation script.

- <*field_name*>_Std
- <*field_name*>_Phon

where <*field_name*> is the name of the field for which you specified person name matching. For example, if you specify the PersonFirstName match type for the **FirstName** field, two fields, **FirstName_Std** and **FirstName_Phon**, are automatically added to the structure.You can also add these fields manually if you do not specify match types in the eView Wizard. If you store additional names in the database, such as alias names, maiden names, parent names, and so on, you can modify the phonetic structure to phonetically encode these additional names as well.

6.2 Match Configuration for Person Data

The default match configuration file, **matchConfigFile.cfg**, defines several match types for the kinds of data typically included in a person master index. You can customize the existing match types or create new match types for the data being processed. The following match types are typical for matching on person data

- FirstName
- MiddleName
- LastName
- String
- Date
- Numeric
- Integer
- Real
- SSN
- Char
- pro
- Exac

This file appears under the **Match Engine** node of the eView Project. For more information about the comparison functions used for each match type and how the weights are tuned, see **"The Match Configuration File" on page 19** and **Appendix B**.

6.3 Standardization Configuration for Person Data

Several configuration files are used to define processing logic for the SeeBeyond Match Engine. You can customize any of the configuration files described in this section to fit your data processing and standardization requirements. These files appear under the **Standardization Engine** node of the eView Project.

6.3.1. personConstants.cfg

The person constants file defines certain information about the standardization files used for processing person data, primarily the number of lines contained in each file. The number of lines specified here must be equal to or greater than the number of lines actually contained in each file.

Table 7 lists and describes each parameter in the constants file. The files referenced by these parameters are described on the following pages.

Table 7 Person Constants File Parameters

Parameter	Description
words	The maximum number of words in a given freeform text field containing a person name. This parameter is not currently used. Default: **11**
conjdex	The maximum number of lines in the person conjunctions reference file (**personConjonUS.dat**). Default: **25**
jrsrdex	The maximum number of lines in the generational suffix category file (**personGenSuffixUS.dat**). Default: **25**
nickdex	The maximum number of lines in the first name category file (**personFirstNameUS.dat**). Default: **3900**
predex	The maximum number of lines in the last name prefix category file (**personLastNamePrefixUS.dat**). Default: **30**
titldex	The maximum number of lines in the title category file (**personTitleUS.dat**). Default: **46**
sufdex	The maximum number of lines in the occupational suffix category file (**personOccupSuffix.dat**). Default: **25**

Table 7 Person Constants File Parameters

Parameter	Description
skpdex	The maximum number of lines in the business name reference file (**businessOrRelated.dat**). Default: **1200**
ptrndex1	The maximum number of lines in the person patterns file (**personNamePatt.dat**). Default: **900**
twodex	The maximum number of lines in the two-character reference file for occupational suffixes (**personTwoUS.dat**). Default: **16**
thredex	The maximum number of lines in the three-character reference file for occupational suffixes (**personTwoUS.dat**). Default: **16**
blnkdex	The maximum number of lines in the special characters reference file (**personRemoveSpecChars.dat**). Default: **16**
dashSize	The maximum number of lines in the hyphenated name category file (**personFirstNameDash.dat**). Default: **20**

6.3.2. personConjonUS.dat

The conjunction reference file is not used in version 5.0, but is designed to work with the person name patterns file during standardization.

6.3.3. personFirstNameDash.dat

The hyphenated name category file defines first names that include hyphens (such as Anne-Marie) to help the SeeBeyond Match Engine recognize and process these values as first names. The file also classifies each name into a gender category.

The syntax of this file is:

```
name gender-class
```

You can modify or add entries in this table as needed. Table 8 describes the columns in the **personFirstNameDash.dat** file.

Table 8 Hyphenated Name Category File

Column	Description
name	A hyphenated first name.
gender-class	An indicator of the gender with which the first name corresponds. The possible values are: ▪ **N**—the title is neutral, and can be applied to male or female first names. ▪ **F**—the title is used for females. ▪ **M**—the title is used for males.

Following is an excerpt from the **personFirstNameDash.dat** file.

```
ANNE-MARIE          F
JEAN-NOEL           M
JEAN-MARIE          M
JEAN-BAPTISTE       M
JEAN-PIERRE         M
JEAN-YVES           M
```

6.3.4. personFirstNameUS.dat

The first name category file defines standardized versions of first names, and defines a gender classification for each name. This file is used to standardize first names when comparing person names. The gender classification helps to further clarify the match. The SeeBeyond Match Engine uses this file when a first name field is defined for normalization or standardization in the Match Field file.

Note: *By default, the names listed in this file are based on US names. You can customize this file to add names of other types, including those containing diacritical marks.*

The syntax of this file is:

```
original-value standardized-form gender-class
```

You can modify or add entries in this table as needed. Table 9 describes the columns in the **personFirstNameUS.dat** file.

Table 9 First Name Category File

Column	Description
original-value	The original value of the first name.
standardized-form	The standardized version of the original value. A zero (0) in this field indicates that the original value is already in its standardized form.
gender-class	An indicator of the gender with which the first name corresponds. The possible values are: ▪ **N**—the title is neutral, and can be applied to male or female first names. ▪ **F**—the title is used for females. ▪ **M**—the title is used for males.

Following is an excerpt from the **personFirstNameUS.dat** file. Certain rows contain a zero (0) for the standardized form, indicating that the name is already standard (Alonzo, Amanda, and Ann, for example).

```
STEPHEN         0               M
STEPHENIE       STEPHANIE       F
STEPHIE         STEPHANIE       F
STEPHINE        STEPHANIE       F
STEPHNIE        STEPHANIE       F
STERLING        0               M
STEVE           STEPHEN         M
STEVEN          STEPHEN         M
STEVIE          STEPHEN         N
STEW            STUART          M
STEWART         STUART          M
STU             STUART          M
STUART          0               M
SU              SUSAN           F
SUE             SUSAN           F
SUHANTO         0               M
SULLIVAN        0               F
SULLY           SULLIVAN        F
SUMMER          0               F
```

6.3.5. personGenSuffixUS.dat

The generational suffix category file defines standardized versions of generational suffixes (such as Jr., III, and so on). This file is used to compare standard versions of the suffix field. You can define additional suffixes and their standardized form, following the syntax below.

```
field-value standard-form
```

Note: *By default, the suffixes listed in this file are based on US names. You can customize this file to add suffixes of other types, including those containing diacritical marks.*

Table 10 describes each column of the **personGenSuffixUS.dat** file.

Table 10 Generational Suffix Category File

Column	Description
field-value	The original value of the generational suffix in the record being processed.
standard-form	The standard form of the generational suffix. A zero ('0') in this column indicates that the value listed in column one is already in its standardized form.

An excerpt from the **personGenSuffixUS.dat** file appears below. In this excerpt, certain suffixes, such as 2ND, 3RD and JR, are already in their standardized form.

```
11          2ND
111         3RD
1V          4TH
2ND         0
3RD         0
4TH         0
FOURTH      4TH
II          2ND
III         3RD
IV          4TH
JR          0
JUNIOR      JR
SECOND      2ND
SENIOR      SR
SR          0
```

6.3.6. personLastNamePrefixUS.dat

The last name prefix category file is not used in version 5.0, but is designed to work with the person name patterns file during standardization.

6.3.7. personNamePatt.dat

The person name patterns file is not used in version 5.0, but is designed to standardize freeform text name fields.

6.3.8. personOccupSuffixUS.dat

The occupational suffix category file is not used in version 5.0, but is designed to work with the person name patterns file during standardization.

6.3.9. personRemoveSpecChars.dat

The special characters reference file lists characters that may appear in person data, but that should be ignored. The SeeBeyond Match Engine removes these characters from a field before making any comparisons or before normalizing data. You can define additional characters to remove from person data by simply adding the character to the list.

An excerpt from the **personRemoveSpecChars.dat** file appears below.

```
[
]
{
}
<
>
/
?
*
^
#
!
```

6.3.10. personThreeUS.dat

This reference file is not used in version 5.0, but is designed to work with the person name patterns file during standardization.

6.3.11. personTitleUS.dat

The title category file defines standard forms for titles and classifies each title into a gender category. For example, "Mister" is standardized to "MR" and is classified as male; "Doctor" is standardized to "DR." and is classified as gender neutral. You can add, modify, or delete entries in this file as needed. Use the following syntax.

```
original-value standardized-form gender-class
```

Note: *By default, the titles listed in this file are based on US names. You can customize this file to add titles of other types, including those containing diacritical marks.*

Table 11 describes each column of the **personTitleUS.dat** file.

Table 11 Person Title Category File

Column	Description
original-value	The original value of the title in the person name field.
standardized-form	The standardized version of the original value. A zero (0) in this field indicates that the original value is already in its standardized form.
gender-class	An indicator of the gender with which the title corresponds. The default values are: ▪ **N**—the title is neither male nor female. ▪ **F**—the title is used for females. ▪ **M**—the title is used for males.

An excerpt from the **personTitleUS.dat** file appears below. In this excerpt, certain titles, such as DR, GEN, and MISS, are already in their standardized form.

```
CTO                         0          N
DEAN                        0          N
DIR                         DIRECTOR   N
DIRECTOR                    0          N
DOC                         DR         N
DOCTOR                      DR         N
DR                          0          N
DRS                         0          N
EMERITUS                    0          N
FOUNDER                     0          N
GEN                         0          N
GENERAL                     GEN        N
MANAGER                     0          N
MGR                         MANAGER    N
MISS                        0          F
MISSUS                      MRS        F
```

6.3.12.personTwoUS.dat

This reference file is not used in version 5.0, but is designed to work with the person name patterns file during standardization.

6.3.13.businessOrRelated.dat

The business-related category file is used to identify business terms in person name information. Examples of when this could occur would be when indexing both person and business names, or when business information is included within a person object structure. The SeeBeyond Match Engine removes these terms for person matching. This file contains a list of common business terms that might be found in person data. You can modify this file by adding, changing, or deleting terms.

An excerpt from the **businessOrRelated.dat** file appears below.

```
ACCOUNTANT
ACCT
ACDY
ACRE
ACREAGE
ACRES
ACS
ACT
AD
ADATU
ADM
ADMIN
ADMINISTRATIO
ADMINISTRATION
ADMINISTRATOR
```

6.4 Customizing Person Data Configuration Files

To customize the SeeBeyond Match Engine configuration files for processing person data, you can modify any of the files described in this chapter using the text editor provided in the Enterprise Designer. Before modifying the match configuration file, review the information provided in **Chapter 4** and **Appendix B** of this guide. Make sure a thorough data analysis has been performed to determine the best fields for matching, and the best comparison functions to use for each field.

Updating most standardization files is a straight-forward process. Make sure to follow the syntax guidelines provided in **"Standardization Configuration for Person Data" on page 38**. If you add any lines to any of the standardization configuration files, be sure to adjust the corresponding parameter in the person constants file (**personConstants.cfg**).

6.5 Configuring the eView Match Service

To ensure the master index uses the SeeBeyond Match Engine to process person information, you must customize the eView Match Service. This includes modifying the Match Field file to support the fields on which you want to match, to standardize the appropriate fields, and to specify the SeeBeyond Match Engine as the match and standardization engine. Perform the following tasks to configure the eView Match Service.

- **Configuring StandardizationConfig** on page 45
- **Configuring the Match String** on page 49
- **Configuring the Match and Standardization Engines** on page 50

When configuring the eView Match Service, keep in mind the information presented in **"Matching Service Configuration" on page 32**.

6.5.1. Configuring StandardizationConfig

The **StandardizationConfig** section of the Match Field file is described in detail in Chapter 6 of the *eView Studio Configuration Guide*. Perform the following steps to configure the required fields for standardization and phonetic encoding.

- **Step 1: Configure the System Object** on page 45
- **Step 2: Configure the Normalization Structures** on page 46
- **Step 3: Configure Fields to Phonetically Encode** on page 48

Note: *In the current configuration, the rules defined for the person data type assume the incoming data to be parsed prior to processing. Therefore, you do not need to configure fields to parse and normalize unless you want to search on address information. In that case, you must configure address fields to parse and normalize, as described in* **"Step 2: Configure Fields to Parse and Normalize" on page 62**.

Step 1: Configure the System Object

The system object defined for standardization must be the parent object of the object structure (typically, this is "Person" for a person database, but it can be whatever you defined).

To configure the system object

1. In the Project Explorer pane of the Enterprise Designer, double-click the **Match Field** node in the **Configuration** folder of the Project you want to modify.
2. Scroll to the **standardize-system-object** element.
3. Make sure the value of the **system-object-name** element is the name of the parent object in the object structure (as defined in the Object Definition file).

   A sample is shown below.

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

4 Save your changes to the file.

Step 2: Configure the Normalization Structures

The fields defined for normalization for the person data type can include any name fields. By default this includes first, middle, and last name fields. The fields are identified by the field identifiers specified for names in **Table 4 on page 28**.

A sample normalization structure for person data is shown below. This sample specifies that the PersonName standardization type is used to normalize the first name, alias first name, last name, and alias last name fields.

```
<structures-to-normalize>
    <group standardization-type="PersonName">
        <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>
    <group standardization-type="LastName">
        <unnormalized-source-fields>
         <source-mapping>
             <unnormalized-source-field-name>Person.LastName
             </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.LastName_Std
             </standardized-target-field-name>
         </target-mapping>
        </normalization-targets>
    </group>
    <group standardization-type="PersonName">
        <unnormalized-source-fields>
         <source-mapping>
             <unnormalized-source-field-name>Person.Alias[*].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.Alias[*].FirstName_Std
```

```
                    </standardized-target-field-name>
                </target-mapping>
               </normalization-targets>
          </group>
          <group standardization-type="LastName">
               <unnormalized-source-fields>
                <source-mapping>
                    <unnormalized-source-field-name>Person.Alias[*].LastName
                    </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[*].LastName_Std
                    </standardized-target-field-name>
                </target-mapping>
               </normalization-targets>
          </group>
     </structures-to-normalize>
```

To configure the normalization structures

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.
3. Add, modify, or delete source or target fields for standardization using the instructions under “Defining Normalization” in Chapter 6 of the *eView Studio Configuration Guide*.

   See Table 12 for guidelines on how to populate the normalization elements for person matching.
4. Save and close the Match Field file.

Table 12 Person Information structures-to-normalize Elements

Element	Description
standardization-type	For name fields, enter “PersonName”. For additional information, see **“Standardization and Match Types” on page 33**.
unnormalized-source-fields elements	
source-mapping	Defines a source field to be normalized.
unnormalized-source-field-name	The ePath of a field in the input record to be normalized. For example, if the parent object is “Person”, and you want to specify the alias last name, the value would be similar to **Person.Alias[*].LastName**.
standardized-object-field-id	The SeeBeyond Match Engine field ID of the field defined by **unnormalized-source-field-name** (for more information, see **Table 4 on page 28**).
normalization-targets elements	
target-mapping	Defines a target field for the corresponding **source-mapping** element.

Table 12 Person Information structures-to-normalize Elements

Element	Description
standardized-object-field-id	The match configuration field ID of the field defined by the **standardized-target-field-name** element. This is the same value as the **standardized-object-field-id** element in the **unnormalized-source-field** element.
standardized-target-field-name	The name of the field that stores the standardized version of the field specified by **unnormalized-source-field-name**. As with **unnormalized-source-field-name**, this value must be the ePath of the field.

Step 3: Configure Fields to Phonetically Encode

When you specify a name field for person name matching in the eView Wizard, these fields are automatically defined for phonetic encoding by default. You can define additional names, such as maiden names or alias names, for phonetic encoding as well. A sample of fields defined for phonetic encoding is shown below. This sample converts name and alias name fields, as well as the street name.

```
<phoneticize-fields>
     <phoneticize-field>
          <unphoneticized-source-field-name>Person.FirstName_StdFirstName
          </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_StdLastName
          </unphoneticized-source-field-name>
          <phoneticized-target-field-name>Person.LastName_Phon
          </phoneticized-target-field-name>
          <encoding-type>NYSIIS</encoding-type>
     </phoneticize-field>
     <phoneticize-field>
          <unphoneticized-source-field-name>Person.Alias[*].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.Alias[*].LastName_Std
          </unphoneticized-source-field-name>
          <phoneticized-target-field-name>Person.LastName_Phon
          </phoneticized-target-field-name>
          <encoding-type>NYSIIS</encoding-type>
     </phoneticize-field>
     <phoneticize-field>
          <unphoneticized-source-field-name>
               Person.Address[*].AddressLine1_StName
          </unphoneticized-source-field-name>
          <phoneticized-target-field-name>
               Person.Address[*].AddressLine1_StPhon
          </phoneticized-target-field-name>
          <encoding-type>NYSIIS</encoding-type>
     </phoneticize-field></phoneticize-fields>
```

To configure fields to phonetically encode

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.

3 Add, modify, or delete fields for phonetic conversion using the instructions under “Defining Phonetic Conversion” in Chapter 6 of the *eView Studio Configuration Guide*.

See Table 13 for guidelines on how to populate the phonetic elements for person matching.

4 Save and close the Match Field file.

Table 13 Person Domain phoneticize-fields Elements

Element	Description
unphoneticized-source-field-name	The ePath of a field in the input record to convert to its phonetic version. For example, if the parent object is “Person” and you want to specify the first name of an alias name, the value would be similar to **Person.Alias[*].FirstName**.
phoneticized-object-field-id	This element is not used by the SeeBeyond Match Engine and can be omitted.
phoneticized-target-field-name	The name of the field that stores the phonetic version of the field. As with the **unphoneticized-source-field-name** element, this value must be the ePath of the field.
encoding-type	The type of encoding to use for the phonetic conversion. Specify “NYSIIS” or “Soundex”.

6.5.2. Configuring the Match String

You can include any fields on which you want to match in the match string. The match string is defined by the **match-column** elements in the **MatchingConfig** section of the Match Field file. If you specify a Match Type for a field in the eView Wizard, that field (or any fields parsed from that field) is automatically defined in the match string.

A sample match string for person matching is shown below. This sample matches on first and last names, date of birth, social security number, gender, and the street name of the address.

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

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

To configure the match string for person matching

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** module.
3. Add, modify, or delete fields in the match string using the instructions under "Configuring the Match String" in Chapter 6 of the *eView Studio Configuration Guide*.

   See Table 14 for guidelines on how to populate the **column-name** and **match-type** elements for person matching.
4. Save and close the Match Field file.

Table 14 Person Domain match-column Elements

Element	Description
column-name	The fully qualified field name of the field in the SBR to be included in the match string, with "Enterprise" as the root object. For example, to specify a person's maiden name, the **column-name** would be similar to **Enterprise.SystemSBR.Person.Maiden**.
match-type	For the SeeBeyond Match Engine, each data type has a different match type. The FirstName, MiddleName, and LastName match types are specific to person matching. You can specify any of the other match types defined in the match configuration file, as well. For more information, see **"Match and Standardization Types" on page 29**.

6.5.3. Configuring the Match and Standardization Engines

The **MEFAConfig** section of the Match Field file defines which standardization and match engines to use based on the adapter and API Java classes specified. Make sure this section is configured for the SeeBeyond Match Engine. Instructions are provided in **"Configuring the Match and Standardization Engines" on page 35**.

For more information, see "MEFA Configuration" in Chapter 6 of the *eView Studio Configuration Guide*.

Chapter 7

Address Data Type Configuration

Processing street addresses involves parsing, normalizing, and phonetically encoding certain fields prior to matching. This chapter describes the configuration files that define address processing logic, and provides instructions for modifying the Match Field file for processing address fields.

7.1 Address Matching Overview

Matching on the address data type includes standardizing and matching on address information in the master index. You can implement street address standardization and matching on its own, or within a master index designed to process person or business information. For example, standardizing address information allows you to include address fields in Enterprise Data Manager search criteria, even though matching might not be performed against these fields.

The SeeBeyond Match Engine can create standardized and phonetic values for United States street address information. Several configuration files designed specifically to handle address data are included to define additional logic for the standardization and phonetic encoding process. These include address clues files, a patterns file, and a constants file. The address standardization engine is based on the work performed at the US Census Bureau. The clues files, in particular, are based on census bureau statistics.

The SeeBeyond Match Engine can match on any field, as long as the match type for the field is defined in the match specifications file (**matchConfigFile.cfg**).

7.1.1. Address Data Processing Fields

When matching on address data, not all fields in a record need to be processed by the SeeBeyond Match Engine. The match engine only needs to process fields that must be parsed, normalized, or phonetically converted, and the fields against which matching is performed. These fields are defined in the Match Field file, and processing logic for each field is defined in the standardization and matching configuration files.

Match String Fields

The match string processed by the SeeBeyond Match Engine is defined by the match fields specified in the Match Field file. If you specify an “Address” match type for any field in the eView Wizard, the parsed address fields are automatically added to the match string in the Match Field file. These fields include the house number, street direction, street type, and street name. You can remove any of these fields from the match string.

The match engine can process any combination of fields you specify for matching. By default, the match specifications file includes rows specifically for matching on the fields that are parsed from the street address fields, such as the street number, street direction, and so on. The file also defines several generic match types. You can use any of the existing rows for matching, or you can create new rows for the fields you want to match.

Standardized Fields

The SeeBeyond Match Engine expects that street address data will be provided in a freeform text field containing several components that must be parsed. The match engine is designed to parse these components, and to normalize and phonetically encode the street name. You can specify additional fields for phonetic encoding.

If you specify an "Address" match type for any field in the eView Wizard, a standardization structure for that field is defined in the Match Field file. The fields listed below under **"The Object Structure"** are automatically defined as the target fields. Each of these fields has several entries in the standardization structure. This is because different parsed components can be stored in the same field. For example, the house number, post office box number, and rural route identifier are all stored in the house number field.

The Object Structure

The address fields specified for standardization are parsed into several additional fields, including one normalized field. If you specify the "Address" match type in the eView Wizard, the following fields are automatically added to the object structure and database creation script.

- <*field_name*>_HouseNo
- <*field_name*>_StName
- <*field_name*>_StDir
- <*field_name*>_StType
- <*field_name*>_StPhon

  where <*field_name*> is the name of the field for which you specified address matching. For example, if you specify the Address match type for the **AddressLine1** field, the following fields are automatically added to the structure: **AddressLine1_HouseNo**, **AddressLine1_StName**, **AddressLine1_StDir**, **AddressLine1_StType**, and **AddressLine1_StPhon**.

You can add these fields manually if you do not specify a match type in the eView Wizard.

7.2 Match Configuration for Address Data

The default match configuration file, **matchConfigFile.cfg**, defines several match types for the kinds of address data typically included in the match string. You can customize the existing match types or create new match types for the data being processed. The following match types are typical for matching on address data.

- StreetNumber
- HouseNumber
- StreetDir
- StreetType

In addition, you can use any of these generic match types for matching on address data.

- String
- Date
- Numeric
- Integer
- Real
- SSN
- Char
- pro
- Exac

This file appears under the **Match Engine** node of the eView Project. For more information about the comparison functions used for each match type and how the weights are tuned, see **"The Match Configuration File" on page 19** and **Appendix B**.

7.3 Standardization Configuration for Address Data

Several configuration files are used to define certain processing logic for the SeeBeyond Match Engine. You can customize any of the configuration files described in this section to fit your data processing and standardization requirements. In addition to the files described here, one additional file, **addressInternalConstants.cfg**, is included. This file is used internally and should not be modified. These files appear under the **Standardization Engine** node of the eView Project.

7.3.1. addressConstants.cfg

The address constants file defines certain information about the standardization files used for processing address data, primarily the number of lines contained in each file. The number of lines specified here must be equal to or greater than the number of lines actually contained in each file.

Table 15 lists and describes each parameter in the constants file. The files referenced by these parameters are described on the following pages.

Table 15 Address Constants File Parameters

Parameter	Description
maxWords	The maximum number of words in a given address field. Default: **15**
clueArraySize	The maximum number of lines in the address clues file (**addressClueAbbrevUS.dat**). Default: **2000**
patternArraySize	The maximum number of lines in the patterns file (**addressPatternsUS.dat**). Default: **1100**
maxPattSize	The maximum length (in characters) of any pattern in the address patterns file. Default: **35**
imageSize	The maximum length of an input address field. Default: **25**
nameOutputFieldSize	The maximum output length of a street or property name. Default: **25**
numberOutputFieldSize	The maximum output length of a house number or rural route number within the structure identifier or post office box fields. Default: **15**
directionOutputFieldSize	The maximum output length of a directional field (prefix or suffix). Default: **15**
typeOutputFieldSize	The maximum output length of a street type field (prefix or suffix). Default: **25**
prefixOutputFieldSize	The maximum length of a number prefix fields. Default: **10**
suffixOutputFieldSize	The maximum length of a number suffix fields. Default: **10**
extensionOutputFieldSize	The maximum output length of any extension field. Default: **10**
extrainfoOutputFieldSize	The maximum output length of any miscellaneous information that is not recognized as a known type. Default: **30**

7.3.2. addressClueAbbrevUS.dat

The address clues file lists common terms in street addresses, specifies a normalized value for each common term, and categorizes the terms into street address component types. A term can be categorized into multiple component types. The relevance value specifies which of the component types the term is most likely to be. For example, the term "Junction" is standardized as "Jct", and is classified as a street type, building unit, and generic term (giving relevance in that order).

This file helps the SeeBeyond Match Engine recognize common terms in street addresses, and to parse and normalize the values correctly. The syntax of this file is:

```
common-term normalized-term ID-number/type-token
```

You can modify or add entries in this table as needed. Table 16 describes the columns in the **addressClueAbbrevUS.dat** file.

Table 16 Address Clues File Columns

Column	Description
common-term	A term commonly found in street addresses.
normalized-term	The normalized version of the common term.
ID-number/type-token	An ID number and a token indicating the type of address component represented by the common term. The ID number corresponds to an ID number in the address master clues file, and the type token corresponds to the type specified for that ID number in the address master clues file. One term might have several ID number and token type pairs.

Following is an excerpt from the **addressClueAbbrevUS.dat** file.

```
TRLR VILLAGE     Trpk         59BU
TRLR VLG         Trpk         59BU
TRPK             Trpk         59BU
TRPRK            Trpk         59BU
VILLA            Vlla         305TY    60BU
VLLA             Vlla         305TY    60BU
VILLAS           Vlla         60BU
VILL             Vlg          317TY    61BU     364AU
VILLAG           Vlg          317TY    61BU     364AU
VLG              Vlg          317TY    61BU     364AU
VILLAGE          Vlg          317TY    61BU     364AU
VILLG            Vlg          317TY    61BU     364AU
VILLIAGE         Vlg          317TY    61BU     364AU
VLGE             Vlg          317TY    61BU     364AU
VIVI             Vivi         62BU
VIVIENDA         Vivi         62BU
COLLEGE          Coll         64BU              0AU
CLG              Coll         64BU
COTTAGE          Cott         65BU     65BP     0AU
```

7.3.3. addressMasterCluesUS.dat

The address master clues file lists common terms in street addresses, as defined by the US Postal Service. For each common term, this file specifies a normalized value, defines USPS information, and categorizes the terms into street address component types. A term can be categorized into multiple component types.

The syntax of this file is:

```
ID-number common-term normalized-term short-abbrev USPS-abbrev CFCCS
type-token usage-flag USPS-flag
```

You can modify or add entries in this table as needed. Table 17 describes the columns in the **addressMasterCluesUS.dat** file.

Table 17 Address Master Clue File Columns

Column	Description
ID-number	A unique identification number for the address common term. This number corresponds to an ID number for the same term in the address clues file.
common-term	A common address term, such as Park, Village, North, and so on.
normalized-term	The normalized version of the common term.
short-abbrev	A short abbreviation of the common term.
USPS-abbrev	The standard United States Postal Office abbreviation of the common term.
CFCCS	The census feature class code of the term (as defined in the Census Tiger® database). The following values are used: ▪ **A**—Road ▪ **B**—Railroad ▪ **C**—Miscellaneous ▪ **D**—Landmark ▪ **E**—Physical feature ▪ **F**—Nonvisible feature ▪ **H**—Hydrography ▪ **X**—Unclassified
type-token	The type of address component represented by the common term. Types are specified by an address token (for more information, see **"Address Type Tokens" on page 58**).
usage-flag	A flag indicating how the term is used (for more information, see **"Pattern Class" on page 60**)
USPS-flag	The United States Postal Office code for the term.

Following is an excerpt from the **addressMasterCluesUS.dat** file.

```
11Alley                 Alley        Al       Aly A    TY R U
12Alternate Route       Alt Rte      Alt      Alt A    TY R
15Arcade                Arcade       Arc      Arc A    TY R U
16Arroyo                Arroyo       Arryo    ArryHA   TY R
17Autopista             Atpta        Apta     AptaA    TY R
18Avenida               Avenida      Ava      Ava A    TY R
19Avenue                Avenue       Ave      Ave A    TY R U
26Boulevard             Blvd         Blvd     BlvdA    TY R U
32Bulevar               Blvr         Blv      Blv A    TY R
33Business Route        Bus Rte      BusRt    BsRtA    TY R
34Bypass                Bypass       Byp      Byp A    TY R U
```

```
36Calle             Calle       Calle    ClleA    TY R
37Calleja           Calleja     Cja      Cja A    TY R
38Callejon          Callej      Cjon     CjonA    TY R
39Camino            Camino      Cam      Cam A    TY R
47Carretera         Carrt       Carr     CarrA    TY R
48Causeway          Cswy        Cswy     CswyAH   TY R U
51Center            Center      Ctr      Ctr DA   TY R U
```

addressPatternsUS.dat

The address patterns file defines the expected input patterns of street address fields to help the SeeBeyond Match Engine recognize and process these values. Tokens are used to indicate the type of address component in the input and output fields. This file contains two rows for each pattern. The first row defines the input pattern and provides an example. The second row defines the output pattern, the pattern type, the relative importance of the pattern compared to other patterns, and usage flags (as shown below).

```
AU A1 TY         01 Oak B Street
NA NA ST         T* 75           TX
```

The relative importance determines which pattern to use in the case that the format of the input field matches more than one pattern.

The syntax of this file is:

```
input-pattern example
output-pattern pattern-class pattern-modifier priority usage-flag
exclude-flag
```

You can modify or add entries in this table as needed. Table 18 describes the columns in the **addressPatternsUS.dat** file.

Table 18 Address Patterns File

Column	Description
input-pattern	Tokens that represent a possible input pattern from the unparsed street address fields. Each token represents one component. For more information about address tokens, see **"Address Type Tokens" on page 58**.
example	An example of a street address that fits the specified pattern. This file element is optional.
output-pattern	Tokens that represent the output pattern for the specified input pattern. Each token represents one component of the output of the SeeBeyond Match Engine. For more information about address tokens, see **"Address Type Tokens" on page 58**.
pattern-class	An indicator of the type of address component represented by the pattern. Possible pattern types are listed in **"Pattern Class" on page 60**.

Table 18 Address Patterns File

Column	Description
pattern-modifier	An indicator of whether the priority of the pattern is averaged against other patterns that match the input. Pattern modifiers are listed in **"Pattern Modifiers" on page 60**.
priority	The priority weight to use for the pattern when the pattern is a sub-pattern of a larger input pattern.
usage-flag	A flag indicating how the term is used (for more information, see **"Pattern Class" on page 60**). This file element is optional.
exclude-flag	This file element is optional.

Following is an excerpt from the **addressPatternsUS.dat** file.

```
NU DR TY A1 AU                 01   123 South Avenida B Oak
HN PD PT NA NA                 H* 70

NU DR TY NU DR                 01   123 South Avenida 1 West
HN PD PT NA SD                 H* 70

NU A1 TY AU TY                 01   123 C circle hill drive
HN HS NA NA ST                 H* 70

NU A1 AM A1 TY                 01   123 M & M road
HN NA NA NA ST                 H* 65

NU TY AU A1                     01   123 Avenida Oak B
HN PT NA NA                    H* 60

NU TY NU A1                    01   123 Avenida 1 B
HN PT NA NA                    H* 60
```

Address Type Tokens

The address pattern and clues files use tokens to denote different components in a street address, such as street type, house number, street names, and so on. These files use one set of tokens for input fields and another set for output fields. You can use only the predefined tokens to represent address components; the SeeBeyond Match Engine does not recognize custom tokens.

Table 19 lists and describes each input token; Table 20 lists and describes each output token.

Table 19 Input Address Pattern Type Tokens

Token	Description
A1	Alphabetic value, one character in length
AM	Ampersand
AU	Generic word
BP	Building property
BU	Building unit

Table 19 Input Address Pattern Type Tokens

Token	Description
BX	Post office box
DA	Dash (as a starting character)
DR	Street direction
EI	Extra information
EX	Extension
FC	Numeric fraction
HR	Highway route
MP	Mile posts
NL	Common words, such as “of”, “the”, and so on
NU	Numeric value
OT	Ordinal type
PT	Prefix type
RR	Rural route
SA	State abbreviations
TY	Street type
WD	Descriptor within the structure
WI	Identifier within the structure

Table 20 Output Address Pattern Tokens

Token	Description
1P	Building number prefix
2P	Second building number prefix
BD	Property or building directional suffix
BI	Structure (building) identifier
BN	Property or building name
BS	Building number suffix
BT	Property or building type suffix
BX	Post office box descriptor
BY	Structure (building) descriptor
DB	Property or building directional prefix
EI	Extra information
EX	Extension index
H1	First house number (the actual number)
H2	Second house number (house number suffix)

Table 20 Output Address Pattern Tokens

Token	Description
HN	House number
HS	House number suffix
NA	Street name
NB	Building number
NL	Conjunctions that connect words or phrases in one component type (usually the street name)
P1	House number prefix
P2	Second house number prefix
PD	Directional prefix to the street name
PT	Street type prefix to the street name
RR	Rural route descriptor
RN	Rural route identifier
ST	Street type suffix to the street name
SD	Directional suffix to the street name
TB	Property or building type prefix
WI	Identifier within the structure
WD	Descriptor within the structure
XN	Post office box identifier

Pattern Class

Each pattern defined in the address patterns file must have an associated pattern class. The pattern class indicates a portion of the input pattern or the type of address data that is represented by the pattern. You can specify any of the following pattern classes.

- **H**—the address pattern represents a house.
- **B**—the address pattern represents a building.
- **W**—the address pattern represents a unit within a structure, such as an apartment or suite number.
- **T**—the address pattern represents a street type or direction.
- **R**—the address pattern represents a rural route.
- **P**—the address pattern represents a Post Office box.
- **N**—the address pattern is mostly numeric.

These classes are also specified as usage flags in the patterns file and the master clues file.

Pattern Modifiers

Each pattern type must be followed by a pattern modifier that indicates how to handle cases where one or more defined patterns is found to be a sub-pattern of a larger input pattern. In this case, the SeeBeyond Match Engine must know how to prioritize each defined pattern that is a part of the larger pattern. The pattern modifiers are:

- *****—An asterisk indicates that the priority weight for the matching pattern is averaged down equally with the other matching sub-patterns.
- **+**—A plus sign indicates that the priority weight for the matching pattern is not averaged down equally with the other matching sub-patterns.

Priority Indicators

The priority indicator is a numeric value following the pattern modifier that indicates the priority weight of the pattern. These values work best when defined as a multiple of five between and including 35 and 95. If a pattern is assigned a priority of 90 or 95 and the pattern matches, or is a sub-pattern of, the input pattern, the match engine stops searching for additional matching patterns and uses the high-priority matching pattern.

7.4 Modifying Address Data Configuration Files

To customize the SeeBeyond Match Engine configuration files for processing street address data, you can modify any of the files described in this chapter using the text editor provided in the Enterprise Designer. Before modifying the match configuration file, review the information provided in **Chapter 4** and **Appendix B** of this guide. Make sure a thorough data analysis has been performed to determine the best fields for matching, and the best comparison functions to use for each field.

Updating most standardization files is a straight-forward process. Make sure to follow the syntax guidelines provided in **"Standardization Configuration for Address Data" on page 53**. If you add rows to any of the standardization files, make sure to adjust the corresponding parameter in the address constants file (**addressConstants.cfg**).

Modifying the patterns file is a more complex task. Only modify this file once you fully understand pattern tokens, types, relevance, and flags.

7.5 Configuring the eView Match Service

To ensure the master index uses the SeeBeyond Match Engine to process address information, you must customize the eView Match Service. This includes modifying the Match Field file to support the fields on which you want to match, to standardize the appropriate fields, and to specify the SeeBeyond Match Engine as the match and standardization engine. Perform the following tasks to configure the eView Match Service.

- **Configuring StandardizationConfig** on page 62
- **Configuring the Match String** on page 65
- **Configuring the Match and Standardization Engines** on page 66

When configuring the eView Match Service, keep in mind the information presented in **"Matching Service Configuration" on page 32**.

7.5.1. Configuring StandardizationConfig

The **StandardizationConfig** section of the Match Field file is described in detail in Chapter 6 of the *eView Studio Configuration Guide*. Perform the following steps to configure the required fields for standardization and phonetic encoding.

- **Step 1: Configure the System Object** on page 62
- **Step 2: Configure Fields to Parse and Normalize** on page 62
- **Step 3: Configure Fields to Phonetically Encode** on page 65

Note: *In the default configuration, the rules defined for the address data type assume that all input fields must be parsed as well as normalized. Thus, this section does not describe the process of configuring fields only for normalization.*

Step 1: Configure the System Object

The system object defined for standardization must be the parent object of the object structure.

To configure the system object

1. In the Project Explorer pane of the Enterprise Designer, double-click the **Match Field** node in the **Configuration** folder of the Project you want to modify.
2. Scroll to the **standardize-system-object** element.
3. Make sure the value of the **system-object-name** element is the name of the parent object in the object structure (as defined in the Object Definition file).

   A sample is shown below.

```
<standardize-system-object>
    <system-object-name>Business</system-object-name>
...
</standardize-system-object>
```

4. Save your changes to the file.

Step 2: Configure Fields to Parse and Normalize

For address fields, the source fields in the standardization structure must include the fields predefined for parsing and normalization. This includes any fields containing street address information, which are parsed into the street address fields listed in **"The Object Structure" on page 52** (except the phonetic street name field). The target fields can include any of these parsed fields.

A sample standardization structure for address data is shown below. This structure parses the first two lines of street address into the standard street address fields.

```
<free-form-texts-to-standardize>
     <group standardization-type="ADDRESS">
          <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>
          <standardization-targets>
           <target-mapping>
               <standardized-object-field-id>HN
               </standardized-object-field-id>
               <standardized-target-field-name>Person.Address[*].HouseNumber
               </standardized-target-field-name>
           </target-mapping>
           <target-mapping>
               <standardized-object-field-id>RV
               </standardized-object-field-id>
               <standardized-target-field-name>Person.Address[*].HouseNumber
               </standardized-target-field-name>
           </target-mapping>
           <target-mapping>
               <standardized-object-field-id>BV
               </standardized-object-field-id>
               <standardized-target-field-name>Person.Address[*].HouseNumber
               </standardized-target-field-name>
           </target-mapping>
           <target-mapping>
               <standardized-object-field-id>SN
               </standardized-object-field-id>
               <standardized-target-field-name>Person.Address[*].StreetName
               </standardized-target-field-name>
           </target-mapping>
           <target-mapping>
               <standardized-object-field-id>RT
               </standardized-object-field-id>
               <standardized-target-field-name>Person.Address[*].StreetName
               </standardized-target-field-name>
           </target-mapping>
           <target-mapping>
               <standardized-object-field-id>BT
               </standardized-object-field-id>
               <standardized-target-field-name>Person.Address[*].StreetName
               </standardized-target-field-name>
           </target-mapping>
           <target-mapping>
               <standardized-object-field-id>PD
               </standardized-object-field-id>
               <standardized-target-field-name>Person.Address[*].StreetDir
               </standardized-target-field-name>
           </target-mapping>
           <target-mapping>
               <standardized-object-field-id>SD
               </standardized-object-field-id>
               <standardized-target-field-name>Person.Address[*].StreetDir
               </standardized-target-field-name>
           </target-mapping>
           <target-mapping>
               <standardized-object-field-id>ST
               </standardized-object-field-id>
               <standardized-target-field-name>Person.Address[*].StreetType
               </standardized-target-field-name>
           </target-mapping>
           <target-mapping>
               <standardized-object-field-id>PT
               </standardized-object-field-id>
               <standardized-target-field-name>Person.Address[*].StreetType
               </standardized-target-field-name>
           </target-mapping>
          </standardization-targets>
     </group>
</free-form-texts-to-standardize>
```

To configure fields to parse and 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 **free-form-text-to-standardize** element.
3. Add, modify, or delete source or target fields for parsing and normalization using the instructions under "Defining Standardization" in Chapter 6 of the *eView Studio Configuration Guide*.

   See Table 21 for guidelines on how to populate the standardization elements for address matching.
4. Save and close the Match Field file.

Table 21 Address free-form-text-to-standardize Elements

Element	Description
standardization-type	Specifies the type of standardization for the SeeBeyond Match Engine to use for the fields defined in the **unstandardized-source-field-name** elements of the group. For parsing street address fields, the value of this element must be "ADDRESS". For more information, see **"Standardization and Match Types" on page 33**.
unstandardized-source-fields elements	
unstandardized-source-field-name	The ePath of a field in the input record containing the freeform text to be standardized. You can have more than one source field (such as AddressLine1 and AddressLine2). The source fields you specify in a group are concatenated to determine the target values for the group. For example, if the parent object is "Business" and you want to specify the first line of the street address, the value would be similar to **Business.Address[*].AddressLine1**.
standardized-targets elements	
target-mapping	Defines a target field for the corresponding **unstandardized-source-fields** element. You can have multiple **target-mapping** elements for one **unstandardized-source-field** element.
standardized-object-field-id	The match configuration field ID of the field defined by the **standardized-target-field-name** element (for more information, see **Table 4 on page 28**)
standardized-target-field-name	The name of the fields that store the standardized version of the input fields. As with the **unstandardized-source-field-name** element, this value must be the ePath of the field.

Step 3: Configure Fields to Phonetically Encode

When you match or stanardize on street address fields, the street name should be specified for phonetic conversion. A sample of the **phoneticize-fields** element is shown below. This sample only converts the address street name. You can define additional fields for phonetic encoding.

```
<phoneticize-fields>
     <phoneticize-field>
          <unphoneticized-source-field-name>Person.Address[*].StreetName
          </unphoneticized-source-field-name>
          <phoneticized-target-field-name>Person.Address[*].StreetName_Phon
          </phoneticized-target-field-name>
          <encoding-type>NYSIIS</encoding-type>
     </phoneticize-field>
</phoneticize-fields>
```

To configure fields to phonetically encode

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.

3 Add, modify, or delete fields for phonetic conversion using the instructions under "Defining Phonetic Conversion" in Chapter 6 of the *eView Studio Configuration Guide*.

  See Table 22 for guidelines on how to populate the phonetic elements for address matching.

4 Save and close the Match Field file.

Table 22 Address phoneticize-fields Elements

Element	Description
unphoneticized-source-field-name	The ePath of a field in the input record to convert to its phonetic version. For example, if the parent object is "Business" and you want to specify the street name, the value would be similar to **Business.Address[*].StreetName**.
phoneticized-object-field-id	This element is not used by the SeeBeyond Match Engine and can be omitted.
phoneticized-target-field-name	The name of the field that stores the phonetic version of the field. As with the **unphoneticized-source-field-name** element, this value must be the ePath of the field.
encoding-type	The type of encoding to use for the phonetic conversion. You can specify "NYSIIS" or "Soundex".

7.5.2. Configuring the Match String

For matching on street address fields, make sure the match string you specify in **MatchingConfig** contains all or a subset of the fields defined for standardization in **StandardizationConfig**. You can include additional fields for matching, such as the city name or zip code. A sample match string for address matching is shown below.

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

To configure the match string for address matching

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** module.
3. Add, modify, or delete fields in the match string using the instructions under "Configuring the Match String" in Chapter 6 of the *eView Studio Configuration Guide*.

   See Table 23 for guidelines on how to populate the **column-name** and **match-type** elements for address matching.
4. Save and close the Match Field file.

Table 23 Address match-column Elements

Element	Description
column-name	The fully qualified field name of the field in the SBR to be included in the match string, with "Enterprise" as the root object. For example, to specify the street name of an address, this value would be similar to **Enterprise.SystemSBR.Business.Address.StreetName**.
match-type	Each component of a street address has a different match type. The default match types are StreetName, HouseNumber, StreetDir, and StreetType. For more information, see **"Match and Standardization Types" on page 29**.

7.5.3. Configuring the Match and Standardization Engines

The **MEFAConfig** section of the Match Field file defines which standardization and match engines to use based on the adapter and API Java classes specified. Make sure this section is configured for the SeeBeyond Match Engine. Instructions are provided in **"Configuring the Match and Standardization Engines" on page 35**.

For more information, see "MEFA Configuration" in Chapter 6 of the *eView Studio Configuration Guide*.

Chapter 8

Business Names Data Type Configuration

Processing unparsed business name fields involves parsing, normalizing, and phonetically encoding certain fields prior to matching. This chapter describes the configuration files that define business name processing logic, and provides instructions for modifying the Match Field file for processing business names.

8.1 Business Name Matching Overview

Matching on the business name data type includes standardizing and matching on freeform business name fields. You can implement business name standardization and matching on its own, or within a master index designed to process person information. For example, standardizing business name fields allows you to include these fields in Enterprise Data Manager search criteria, even though matching might not be performed against these fields.

The SeeBeyond Match Engine can create standardized and phonetic values for business names. Several configuration files designed specifically to handle business names are included to define additional logic for the standardization and phonetic encoding process. These include reference files, a patterns file, and key type files. The SeeBeyond Match Engine can match on any field, as long as the match type for the field is defined in the match specifications file (**matchConfigFile.cfg**).

8.1.1. Business Name Processing Fields

When matching on freeform business names, not all fields in a record need to be processed by the SeeBeyond Match Engine. The match engine only needs to process fields that must be parsed, normalized, or phonetically converted, and the fields against which matching is performed. These fields are defined in the Match Field file, and processing logic for each field is defined in the standardization and matching configuration files.

Match String Fields

The match string processed by the SeeBeyond Match Engine is defined by the match fields specified in the Match Field file. If you specify a “BusinessName” match type for any field in the eView Wizard, most of the parsed business name fields are automatically added to the match string in the Match Field file. The fields added to the match string include the name, organization type, association type, sector, industry, and URL. You can remove any of these fields from the match string.

MatchThe match engine can process any combination of fields you specify for matching. By default, the match specifications file includes rows specifically for matching on the fields that are parsed from the business name fields, such as the name, industry, and so on. The file also defines several generic match types. You can use any of the existing rows for matching, or you can create new rows for the fields you want to match.

Standardized Fields

The SeeBeyond Match Engine expects that business name data will be provided in a freeform text field containing several components that must be parsed. The match engine is designed to parse these components, and to normalize and phonetically encode the business name. You can specify additional fields for phonetic encoding.

If you specify the "BusinessName" match type for any field in the eView Wizard, a standardization structure for that field is defined in the Match Field file. The fields defined as the target fields are listed in the next section, **"The Object Structure"**.

The Object Structure

For the default configuration of the business name data type, the address fields specified for standardization are parsed into several additional fields, including one normalized field. If you specify the appropriate match type in the eView Wizard, the following fields are automatically added to the object structure and database creation script.

- <*field_name*>_Name
- <*field_name*>_NamePhon
- <*field_name*>_OrgType
- <*field_name*>_AssocType
- <*field_name*>_Industry
- <*field_name*>_Sector
- <*field_name*>_Alias
- <*field_name*>_Url

  where <*field_name*> is the name of the field for which you specified business name matching. For example, if you specify the BusinessName match type for the **Company** field, the fields automatically added to the structure include **Company_Name**, **Company_NamePhon**, **Company_OrgType**, and so on.

You can add these fields manually if you do not specify a match type in the eView Wizard.

8.2 Match Configuration for Business Names

The default match configuration file, **matchConfigFile.cfg**, defines several match types for the kinds of business name data typically included in the match string. You can customize the existing match types or create new match types for the data being processed. The following match types are typical for matching on business names.

- PrimaryName
- OrgTypeKeyword
- AssocTypeKeyword
- IndustrySectorList
- AliasList
- IndustryTypeKeyword
- URL

In addition, you can use any of these generic match types for matching on business names.

- String
- Date
- Numeric
- Integer
- Real
- Char
- pro
- Exac

This file appears under the **Match Engine** node of the eView Project. For more information about the comparison functions used for each match type and how the weights are tuned, see **"The Match Configuration File" on page 19** and **Appendix B**.

8.3 Standardization Configuration for Business Names

Several configuration files are used to define certain processing logic for the SeeBeyond Match Engine. You can customize any of the configuration files described in this section to fit your data processing and standardization requirements. These files appear under the **Standardization Engine** node of the eView Project.

8.3.1. bizConstants.cfg

The business constants file defines certain information about the standardization files used for processing business data, primarily the number of lines contained in each file. The number of lines specified here must be equal to or greater than the number of lines actually contained in each file.

Table 24 lists and describes each parameter in the constants file. The files referenced by these parameters are described on the following pages.

Table 24 Business Constants File Parameters

Parameter	Description
cityMax	The maximum number of lines in the city or state key type file (**bizCityorStateTypeKey.dat**). Default: **180**
primaryMax	The maximum number of lines in the primary business names reference file (**bizCompanyPrimaryNames.dat**). Default: **525**
countryMax	The maximum number of lines in the country key type file (**bizCountryTypeKeys.dat**). Default: **99**
industryMax	The maximum number of lines in the industry key type file (**bizIndustryTypeKeys.dat**). Default: **900**
patternMax	The maximum number of lines in the business patterns file (**bixPatterns.dat**). Default: **400**
mergerMax	The maximum number of lines in the merged business name category file (**bizCompanyMergerNames.dat**). Default: **40**
adjectiveMax	The maximum number of lines in the adjective key type file (**bizAdjectiveTypeKeys.dat**). Default: **80**
orgMax	The maximum number of lines in the organization key type file (**bizOrganizationTypeKeys.dat**). Default: **50**
assocMax	The maximum number of lines in the association key type file (**bizAssociationTypeKeys.dat**). Default: **20**
genTermMax	The maximum number of lines in the general terms reference file (**bizBusinessGeneralTerms.dat**). Default: **20**
charsMax	The maximum number of lines in the special characters reference file (**bizRemoveSpecChars.dat**). Default: **25**

8.3.2. bizAdjectivesTypeKeys.dat

The adjectives key type file defines adjectives commonly found in business names to help the SeeBeyond Match Engine recognize and process these values as a part of the business name. This file contains one column with a list of commonly used adjectives, such as General, Financial, Central, and so on.

You can modify or add entries in this file as needed. Following is an excerpt from the **bizAdjectivesTypeKeys.dat** file.

```
DIGITAL
DIRECTED
DIVERSIFIED
EDUCATIONAL
ELECTROCHEMICAL
ENGINEERED
EVOLUTIONARY
EXTENDED
FACTUAL
FEDERAL
```

8.3.3. bizAliasTypeKeys.dat

The alias key type file lists business name acronyms and abbreviations along with their standardized names. This helps the SeeBeyond Match Engine recognize and process these values appropriately. You can add entries to the alias key type file using the following syntax.

```
alias standardized-name
```

Table 25 describes the columns in the **bizAliasTypeKeys.dat** file.

Table 25 Alias Key Type File

Column	Description
alias	An abbreviation or acronym commonly used in place of a specific business name.
standardized-name	The normalized version of the alias name.

Following is an excerpt from the **bizAliasTypeKeys.dat** file.

```
BBH                     BARTLE BOGLE HEGARTY
BBH                     BROWN BROTHERS HARRIMAN
IBM                     INTERNATIONAL BUSINESS MACHINE
IDS                     INCOMES DATA SERVICES
IDS                     INSURANCE DATA SERVICES
IDS                     THE INTEGRATED DECISION SUPPORT GROUP
IDS                     THE INTERNET DATABASE SERVICE
CAL-TECH                CALIFORNIA INSTITUTE OF TECHNOLOGY
```

8.3.4. bizAssociationTypeKeys.dat

The association key type file lists business association types along with their standardized names. This helps the SeeBeyond Match Engine recognize and process these values appropriately. You can add entries to the association key type file using the following syntax.

```
association-type standardized-type
```

Table 26 describes the columns in the **bizAssociationTypeKeys.dat** file.

Table 26 Association Type Key Table

Column	Description
association-type	A common association type for businesses, such as Partners, Group, and so on.

Table 26 Association Type Key Table

Column	Description
standardized-type	The standardized version of the association type.

Following is an excerpt from the **bizAssociationTypeKeys.dat** file.

```
ASSOCIATES          0
BANCORP             0
BANCORPORATION      BANCORP
COMPANIES           0
GP                  GROUP
GROUP               0
PARTNERS            0
```

8.3.5. bizBusinessGeneralTerms.dat

The general terms reference file lists terms commonly used in business names. This file is used to identify terms that indicate a business, such as bank, supply, factory, and so on. This helps the SeeBeyond Match Engine recognize and process the business name.

This file contains one column that lists the connector tokens in the business names you process. You can add entries as needed. Below is an excerpt from the **bizBusinessGeneralTerms.dat** file.

```
BUILDING
CITY
CONSUMER
EAST
EYE
FACTORY
LATIN
NORTH
SOUTH
```

8.3.6. bizCityorStateTypeKeys.dat

The city or state key type file lists various cities and states that might be used in business names. It also classifies each entry as a city (CT) or state (ST) and indicates the country in which the city or state is located. This helps the SeeBeyond Match Engine recognize and process these values appropriately. You can add entries to the city or state key type file using the following syntax.

```
city-or-state type country
```

Table 26 describes the columns in the **bizCityorStateTypeKeys.dat** file.

Table 27 City or State Key Type File

Column	Description
city-or-state	The name of a city or state used in business names.
type	An indicator of whether the value is a city or state. "CT" indicates city and "ST" indicates state.

Table 27 City or State Key Type File

Column	Description
country	The country code of the country in which the city or state is located.

Following is an excerpt from the **bizCityorStateTypeKeys.dat** file.

```
ADELAIDE                 CT   AU
ALABAMA                  ST   US
ALASKA                   ST   US
ALGIERS                  CT   DZ
AMSTERDAM                CT   NL
ARIZONA                  ST   US
ARKANSAS                 ST   US
ASUNCION                 CT   PY
ATHENS                   CT   GR
```

8.3.7. bizCompanyFormerNames.dat

The business former name reference file provides a list of common company names along with names by which the companies were formerly known. This helps the SeeBeyond Match Engine recognize a business when a record containing their previous business name is processed. You can add entries to the business former name table using the following syntax.

```
former-name current-name
```

Table 28 describes each column in the **bizCompanyFormerNames.dat** file.

Table 28 Business Former Name Reference File

Column	Description
former-name	One of the company's previous names.
current-name	The company's current name.

Below is an excerpt from the **bizCompanyFormerNames.dat** file.

```
HELLENIC BOTTLING                       COCA-COLA HBC
INTERNATIONAL PRODUCTS                  THE TERLATO WINE
ORGANIC FOOD PRODUCTS                   SPECTRUM ORGANIC PRODUCTS
SOFTWARE TECHNOLOGY CORPORATION         SEEBEYOND TECHNOLOGY CORPORATION
SUTTER HOME WINERY                      TRINCHERO FAMILY ESTATES
```

8.3.8. bizCompanyMergerNames.dat

The merged business name category file provides a list of companies whose name changed because of a merger along with the name of the company after the merge. It also classifies the business names into industry sectors and sub-sectors. This helps the SeeBeyond Match Engine recognize the current company name and to determine the sector of the business. You can add entries to the business merger name file using the following syntax.

```
former-name/merged-name sector-code
```

Table 28 describes each column in the **bizCompanyMergerNames.dat** file.

Table 29 Business Merger Name Category File

Column	Description
former-name	The name of the company whose name was not kept after the merger.
merged-name	The name of the company whose name was kept after the merger.
sector-code	The industry sector code of the business. Sector codes are listed in the **bizIndustryCategoriesCode.dat** file.

Below is an excerpt from the **bizCompanyMergerNames.dat** file.

```
DUKE/FLUOR DANIEL                                       20005
FAULTLESS STARCH/BON AMI                                09004
FIND/SVP                                                10013
FIRST WAVE/NEWPARK SHIPBUILDING                         27005
GUNDLE/SLT                                              19020
HMG/COURTLAND                                           23004
J BROWN/LMC                                             10014
KORN/FERRY                                              10020
LINSCO/PRIVATE LEDGER                                   14005
```

8.3.9. bizCompanyPrimaryNames.dat

The primary business name reference file provides a list of companies by their primary name. It also classifies the business names into industry sectors and sub-sectors. This helps the SeeBeyond Match Engine determine the correct value of the sector field when parsing the business name. You can add entries to the primary business name file using the following syntax.

```
primary-name sector-code
```

Table 28 describes the columns in the **bizCompanyPrimaryNames.dat** file.

Table 30 Business Primary Name Reference File

Column	Description
primary-name	The primary name of the company.
sector-code	The industry sector code of the business. Sector codes are listed in the **bizIndustryCategoriesCode.dat** file.

Below is an excerpt from the **bizCompanyPrimaryNames.dat** file.

```
BROTHER INTERNATIONAL                             12006
BRYSTOL-MYERS SQUIBB                              11005
BURLINGTON COAT FACTORY                           24003
BURLINGTON NORTHERN SANTA FE                      27005
BV SOLUTIONS                                      06012
CABLEVISION                                       26001
CABOT                                             04006
CADENCE                                           06010
CAMPBELL                                          22006
CAPITAL BLUE CROSS                                17001
```

8.3.10. bizConnectorTokens.dat

The connector tokens reference file defines common values (typically conjunctions) that connect words in business names. For example, in the business name “Nursery of Venice”, “of” is a connector token. This helps the SeeBeyond Match Engine recognize and process the full name of a business by indicating that the token connects two parts of the full name.

This file contains one column that lists the connector tokens in the business names you process. You can add entries as needed. Below is an excerpt from the **bizConnectorTokens.dat** file.

```
AN
DE
DES
DOS
LA
LAS
LE
OF
THE
```

8.3.11. bizCountryTypeKeys.dat

The country key type file lists countries and continents, along with their abbreviations and assigned nationalities. For continents, the abbreviation is “CON” to separate them from countries. This helps the SeeBeyond Match Engine recognize and process these values as countries or continents. You can add entries to the country key type file using the following syntax.

```
country abbreviation nationality
```

Table 26 describes the columns in the **bizCountryTypeKeys.dat** file.

Table 31 Country Key Type Files

Column	Description
country	The name of a country or continent.
abbreviation	The common abbreviation for the specified country. The abbreviation for a continent is always “CON”.
nationality	The nationality assigned to a person or business originating in the specified country.

Following is an excerpt from the **bizCountryTypeKeys.dat** file.

```
AMERICA                        CON  AMERICAN
AFRICA                         CON  AFRICAN
EUROPE                         CON  EUROPEAN
ASIA                           CON  ASIAN
AFGHANISTAN                    AF   AFGHAN
ALBANIA                        AL   ALBANIAN
ALGERIA                        DZ   ALGERIAN
```

8.3.12. bizIndustryCategoryCode.dat

The industry sector reference file lists and groups various industry sectors and sub-sectors, and includes an identification code for each type. This helps the SeeBeyond Match Engine determine and process the industry sectors for different businesses. You can add entries to the industry sector reference file using the following syntax.

```
sector-code industry-sector
```

Table 33 describes each column in the **bizIndustryCategoryCode.dat** file.

Table 32 Industry Sector Reference File

Column	Description
sector-code	The identification code of the specified sector. The first two numbers of each code identify the general industry sector; the last three number identify a sub-sector.
industry-sector	A description of the industry category. This is written in the format “<sector> - <sub-sector>”, where <sector> is a general category of industry types, and <sub-sector> is a specific industry within that category.

```
02006    Automotive & Transport Equipment - Recreational Vehicles
02007    Automotive & Transport Equipment - Shipbuilding & Related Services
02008    Automotive & Transport Equipment - Trucks, Buses & Other Vehicles
03001    Banking - Banking
04001    Chemicals - Agricultural Chemicals
04002    Chemicals - Basic & Intermediate Chemicals & Petrochemicals
04003    Chemicals - Diversified Chemicals
04004    Chemicals - Paints, Coatings & Other Finishing Products
04005    Chemicals - Plastics & Fibers
04006    Chemicals - Specialty Chemicals
05001    Computer Hardware - Computer Peripherals
05002    Computer Hardware - Data Storage Devices
05003    Computer Hardware - Diversified Computer Products
```

8.3.13. bizIndustryTypeKeys.dat

The industry key type file is used to standardize the value of the Industry field into common industries to which businesses belong. This helps the SeeBeyond Match Engine recognize and process the industry types for different businesses. You can add entries to the industry key type file using the following syntax.

```
industry-type standardized-form sectors
```

Table 33 describes each column in the **bizIndustryTypeKeys.dat** file.

Table 33 Industry Key Type File

Column	Description
industry-type	The original value of the industry type in the input record.
standardized-form	The normalized version of the industry type.
sectors	The industry categories of the specified industry type. These values correspond to the sector codes listed in the industry sector file (**bizIndustryCategoryCode.dat**). You can list as many categories as apply for each type, but they must be entered with a space between each and no line breaks, and they must correspond to an entry in the industry sector file.

Below is an excerpt from the **bizIndustryTypeKeys.dat** file.

```
TECH                  TECHNOLOGY          05001-05007
TECHNOLOGIES          TECHNOLOGY          05001-05007
TECHNOLOGY            0                   05001-05007
TECHSYSTEMS           0                   05001-05007
TELE PHONE            TELEPHONE           16005
TELE PHONES           TELEPHONES          16005
TELEVISION            TV                  11013  21014
TELECOM               0                   16005  26006  26009  26010
TELECOMM              TELECOMMUNICATION   16005  26006  26008
TELECOMMUNICATION     0                   16005  26006  26008
```

8.3.14. bizOrganizationTypeKeys.dat

The organization key type file is used to standardize the value of the Organization field into common organizations to which businesses belong. This helps the SeeBeyond Match Engine recognize and process the organization types for different businesses. You can add entries to the organization key type file using the following syntax.

```
original-type standardized-form
```

Table 34 describes each column in the **bizOrganizationTypeKeys.dat** file.

Table 34 Organization Key Type File

Column	Description
original-type	The original value of the organization field in an input record.
standardized-form	The normalized version of an organization type. A zero (0) in this field indicates that the value in the first column is already in its standardized form.

Below is an excerpt from the **bizOrganizationTypeKeys.dat** file.

```
INC                 INCORPORATED
INCORPORATED        0
KG                  0
KK                  0
LIMITED             0
LIMITED PARTNERSHIP 0
LLC                 0
LLP                 0
LP                  LIMITED PARTNERSHIP
LTD                 LIMITED
```

8.3.15.bizPatterns.dat

The business patterns file defines multiple formats expected from the business name input fields along with the standardized output of each format. The patterns and output appear in two-row pairs in this file, as shown below.

```
4 PNT AST SEP-GLC ORT
PNT AST DEL ORT
```

The first line describes the input pattern and the second line describes the output pattern. This file uses tokens to denote each component. The allowed tokens are described in **"Business Name Tokens" on page 80**. A number at the beginning of the first line indicates the number of components in the given business name format. You can modify this file using the following syntax.

```
length input-pattern
output-pattern
```

Table 35 lists and describes the syntax components.

Table 35 Business Patterns File Components

Component	Description
length	The number of business name components in the input field.
input-pattern	Tokens that represent a possible input pattern from the unparsed business name fields. Each token represents one component. For more information about address tokens, see **"Business Name Tokens" on page 80**.
output-pattern	Tokens that represent the output pattern for the specified input pattern. Each token represents one component. For more information about business name tokens, see **"Business Name Tokens" on page 80**.

Below is an excerpt from the **bizPatterns.dat** file.

```
4 PNT AST SEP-GLC ORT
PNT AST DEL ORT

4 PNT IDT IDT ORT
PNT IDT ORT

4 PNT IDT-AJT IDT ORT
PNT IDT IDT ORT

4 NF IDT-AJT IDT ORT
PNT IDT IDT ORT

4 AJT IDT SEP-GLC ORT
PNT PNT DEL ORT
```

Business Name Tokens

The business patterns file uses tokens to denote different components in a business name, such as the primary name, alias type key, URL, and so on. These files use one set of tokens for input fields and another set for output fields. The tokens indicate the type key files to use to determine the appropriate values for each output field. You can use only the predefined tokens to represent business name components; the SeeBeyond Match Engine does not recognize custom tokens.

Table 36 lists and describes each input token; Table 37 lists and describes each output token.

Table 36 Business Name Input Pattern Tokens

Pattern Identifier	Description
CTT	A connector token
PNT	A primary name of a business
PN-PN	A hyphenated primary name of a business
BCT	A common business term
URL	The URL of the business' web site
ALT	A business alias type key (usually an acronym)
CNT	A country name
NAT	A nationality
CST	A city or state type key
IDT	An industry type key
IDT-AJT	Both an industry and an adjective type key
AJT	An adjective type key
AST	An association type key
ORT	An organization type key.
SEP	A separator key
NFG	Generic term, not recognized as a specific business name component, with an internal hyphen

Table 36 Business Name Input Pattern Tokens

Pattern Identifier	Description
NF	Generic term, not recognized as a specific business name component
NFC	A single character, not recognized as a specific business name component
SEP-GLC	A joining comma (a *glue type separator*)
SEP-GLD	A joining hyphen (a *glue type separator*)
AND	The text "and"
GLU	A glue type key, such as a forward slash, connecting two parts of a business name component
PN-NF	A business primary name followed by a hyphen and a generic term that is not recognized as a specific business name component
NF-PN	A generic term that is not recognized as a specific business name component, followed by a hyphen and a recognized business primary name
NF-NF	Two generic terms, not recognized as specific business name components and separated by a hyphen

Table 37 Business Name Output Pattern Tokens

Pattern Identifier	Description
PNT	The primary name of the business
URL	The URL of the business
ALT	The alias type key of the business (usually an acronym)
IDT	The industry type key of the business
AST	The association type key of the business
ORT	The organization type key of the business
NF	A generic term not recognized as a business name component

8.3.16.bizRemoveSpecChars.dat

The special characters reference file lists certain characters that should be removed from a business name prior to processing the field. These typically include punctuation marks such as exclamation points, parenthesis, and so on. This helps the SeeBeyond Match Engine recognize the business name.

This file contains one column that lists the characters to be removed from the business names you process. You can add entries as needed. Below is an excerpt from the **bizRemoveSpecChars.dat** file.

```
[
]
{
}
<
>
/
?
```

8.4 Modifying Business Name Configuration Files

To customize the SeeBeyond Match Engine configuration files for processing business names, you can modify any of the files described in this chapter using the text editor provided in the Enterprise Designer. Before modifying the match configuration file, review the information provided in **Chapter 4** and **Appendix B** of this guide. Make sure a thorough data analysis has been performed to determine the best fields for matching, and the best comparison functions to use for each field.

Updating most standardization files is a straight-forward process. Make sure to follow the syntax guidelines provided in **"Standardization Configuration for Business Names" on page 70**. If you add rows to any standardization files, make sure to modify the corresponding parameter in the business constants file (**bizConstants.cfg**). Before making any changes to the patterns file, make sure you understand the tokens used to represent business name field components.

8.5 Configuring the eView Match Service

To ensure the master index uses the SeeBeyond Match Engine to process business names, you must customize the eView Match Service. This includes modifying the Match Field file to support the fields on which you want to match, to standardize the appropriate fields, and to specify the SeeBeyond Match Engine as the match and standardization engine. Perform the following tasks to configure the eView Match Service.

- **Configuring StandardizationConfig** on page 83
- **Configuring the Match String** on page 86
- **Configuring the Match and Standardization Engines** on page 87

When configuring the eView Match Service, keep in mind the information presented in **"Matching Service Configuration" on page 32**.

8.5.1. Configuring StandardizationConfig

The **StandardizationConfig** section of the Match Field file is described in detail in Chapter 6 of the *eView Studio Configuration Guide*. Perform the following steps to configure the required fields for standardization and phonetic encoding.

- **Step 1: Configure the System Object** on page 83
- **Step 2: Configure Fields to Parse and Normalize** on page 83
- **Step 3: Configure Fields to Phonetically Encode** on page 85

Note: *In the default configuration, the rules defined for the business data type assume that all input fields must be parsed as well as normalized. Thus, this section does not describe the process of configuring fields only for normalization.*

Step 1: Configure the System Object

The system object defined for standardization must be the parent object of the object structure.

To configure the system object

1. In the Project Explorer pane of the Enterprise Designer, double-click the **Match Field** node in the **Configuration** folder of the Project you want to modify.
2. Scroll to the **standardize-system-object** element.
3. Make sure the value of the **system-object-name** element is the name of the parent object in the object structure (as defined in the Object Definition file).

   A sample is shown below.

```
<standardize-system-object>
    <system-object-name>Company</system-object-name>
...
</standardize-system-object>
```

4. Save your changes to the file.

Step 2: Configure Fields to Parse and Normalize

For business name fields, the source fields in the standardization structure must include the fields predefined for parsing and normalization. This includes any fields containing business name information, which are parsed into the business name fields listed in **"The Object Structure" on page 69** (except the phonetic business name field). The target fields can include any of these parsed fields.

A sample standardization structure for business name data is shown below. This structure parses a business name field into the standard business name fields.

```
<free-form-texts-to-standardize>
    <group standardization-type="BusinessName">
        <unstandardized-source-fields>
         <unstandardized-source-field-name>Company.Name
         </unstandardized-source-field-name>
        </unstandardized-source-fields>
        <standardization-targets>
         <target-mapping>
```

```
                    <standardized-object-field-id>PrimaryName
                    </standardized-object-field-id>
                    <standardized-target-field-name>Company.Name_Name
                    </standardized-target-field-name>
                </target-mapping>
                <target-mapping>
                    <standardized-object-field-id>OrgTypekeyword
                    </standardized-object-field-id>
                    <standardized-target-field-name>Company.Name_OrganizationType
                    </standardized-target-field-name>
                </target-mapping>
                <target-mapping>
                    <standardized-object-field-id>AssocTypeKeyword
                    </standardized-object-field-id>
                    <standardized-target-field-name>Company.Name_AssociationType
                    </standardized-target-field-name>
                </target-mapping>
                <target-mapping>
                    <standardized-object-field-id>IndustrySectorList
                    </standardized-object-field-id>
                    <standardized-target-field-name>Company.Name_Sector
                    </standardized-target-field-name>
                </target-mapping>
                <target-mapping>
                    <standardized-object-field-id>IndustryTypeKeyword
                    </standardized-object-field-id>
                    <standardized-target-field-name>Company.Company.Name_Industry
                    </standardized-target-field-name>
                </target-mapping>
                <target-mapping>
                    <standardized-object-field-id>AliasList
                    </standardized-object-field-id>
                    <standardized-target-field-name>Company.Company.Name_Alias
                    </standardized-target-field-name>
                </target-mapping>
                <target-mapping>
                    <standardized-object-field-id>Url
                    </standardized-object-field-id>
                    <standardized-target-field-name>Company.Company.Name_URL
                    </standardized-target-field-name>
                </target-mapping>
               </standardization-targets>
          </group>
     </free-form-texts-to-standardize>
```

To configure fields to parse and 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 **free-form-text-to-standardize** element.
3. Add, modify, or delete source or target fields for parsing and normalization using the instructions under "Defining Standardization" in Chapter 6 of the *eView Studio Configuration Guide*.

   See Table 38 for guidelines on how to populate the standardization elements for business name matching.

4 Save and close the Match Field file.

Table 38 Business Name Domain free-form-text-to-standardize Elements

Element	Description
standardization-type	Specifies the type of standardization to perform on the fields defined in the **unstandardized-source-field-name** elements of the group. For parsing business name fields, the value of this element must be "BusinessName". For more information, see **"Standardization and Match Types" on page 33**.
unstandardized-source-fields elements	
unstandardized-source-field-name	The ePath of a field in the input record containing the free-form text to be standardized. You can specify more than one source field. The source fields you specify in a group are concatenated to determine the target values for the group. For example, if the parent object is "Company" and you want to specify the industry type, the value would be similar to **Company.Industry_Type**.
standardized-targets elements	
target-mapping	Defines a target field for the corresponding **unstandardized-source-fields** element.
standardized-object-field-id	The match configuration field ID of the field defined by the **standardized-target-field-name** element (for more information, see **Table 4 on page 28**).
standardized-target-field-name	The name of the fields that store the standardized version of the input fields. As with the **unstandardized-source-field-name** element, this value must be the ePath of the field.

Step 3: Configure Fields to Phonetically Encode

When you match on business name fields, the name field should be specified for phonetic conversion. A sample of the **phoneticize-fields** element is shown below. This sample only converts the business name. You can define additional fields for phonetic encoding.

```
<phoneticize-fields>
     <phoneticize-field>
          <unphoneticized-source-field-name>Company.Name_Name
          </unphoneticized-source-field-name>
          <phoneticized-target-field-name>Company.Name_NamePhon
          </phoneticized-target-field-name>
          <encoding-type>NYSIIS</encoding-type>
     </phoneticize-field>
</phoneticize-fields>
```

To configure fields to phonetically encode

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.

3 Add, modify, or delete fields for phonetic conversion using the instructions under "Defining Phonetic Conversion" in Chapter 6 of the *eView Studio Configuration Guide*.

See Table 39 for guidelines on how to populate the phonetic elements for business name matching.

4 Save and close the Match Field file.

Table 39 Business Name Data Type phoneticize-fields Elements

Element	Description
unphoneticized-source-field-name	The ePath of a field in the input record to convert to its phonetic version. For example, if the parent object is "Company" and you want to specify the company name, the value would be similar to **Company.Company_Name**.
phoneticized-object-field-id	This element is not used by the SeeBeyond Match Engine and can be omitted.
phoneticized-target-field-name	The name of the field that will store the phonetic version of the field. As with the **unphoneticized-source-field-name** element, this value must be the ePath of the field.
encoding-type	The type of encoding to use for the phonetic conversion. You can specify "NYSIIS" or "Soundex".

8.5.2. Configuring the Match String

For matching on business name fields, make sure the match string you specify in **MatchingConfig** contains all or a subset of the fields defined for standardization in **StandardizationConfig**. You can include additional fields for matching if required. A sample match string for business name matching is shown below. This sample matches on the company name, the organization type, and the sector.

```
<match-system-object>
     <object-name>Company</object-name>
     <match-columns>
          <match-column>
           <column-name>Enterprise.SystemSBR.Company.Name_PrimaryName
           </column-name>
           <match-type>PrimaryName</match-type>
          </match-column>
          <match-column>
           <column-name>Enterprise.SystemSBR.Company.Name_OrganizationType
           </column-name>
           <match-type>OrgTypeKeyword</match-type>
          </match-column>
          <match-column>
           <column-name>Enterprise.SystemSBR.Company.Name_Sector
           </column-name>
           <match-type>IndustryTypeKeyword</match-type>
          </match-column>
     </match-columns>
</match-system-object>
```

To configure the match string for address matching

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** module.

3 Add, modify, or delete fields in the match string using the instructions under "Configuring the Match String" in Chapter 6 of the *eView Studio Configuration Guide*.

See Table 40 for guidelines on how to populate the **column-name** and **match-type** elements for business name matching.

4 Save and close the Match Field file.

Table 40 Business Name Data Type match-column Elements

Element	Description
column-name	The fully qualified field name of the field in the SBR to be included in the match string, with "Enterprise" as the root object. For example, to specify the company name, the **column-name** would be similar to **Enterprise.SystemSBR.Company.CompanyName**.
match-type	Each component of a business name has a different match type. The default match types are PrimaryName, OrgTypeKeyword, AssocTypeKeyword, IndustrySectorList, IndustryTypeKeyword, and Url. For more information, see **"Match and Standardization Types" on page 29**.

8.5.3. Configuring the Match and Standardization Engines

The **MEFAConfig** section of the Match Field file defines which standardization and match engines to use based on the adapter and API Java classes specified. Make sure this section is configured for the SeeBeyond Match Engine. Instructions are provided in **"Configuring the Match and Standardization Engines" on page 39**.

For more information, see "MEFA Configuration" in Chapter 6 of the *eView Studio Configuration Guide*.

Appendix A

Fine-tuning Weights and Thresholds

Each eView implementation is unique, typically requiring extensive data analysis to determine how to best configure the structure and matching logic of the master index. This chapter provides an overview of the process of fine-tuning the matching logic in the match configuration file and fine-tuning the match and duplicate thresholds.

1.1 Overview

A thorough analysis of the data to be shared with the master index is a must before beginning any implementation. This analysis not only defines the types of data to include in the structure, but indicates the relative reliability of each system’s data, helps determine which fields should be used for matching, and should indicate the relative reliability of each match field.

To begin the analysis, the legacy data that will be converted into the database should be extracted and analyzed. Once the initial analysis is complete, you can perform an iterative process to help fine-tune the matching and duplicate thresholds, and to determine the level of potential duplication in the existing data.

1.2 Customizing the Match Configuration and Thresholds

There are three primary steps to customizing the way records are matched in the master index. These tasks are described on the following pages.

- **Determine the Match Fields** on page 88
- **Customize the Match Configuration** on page 89
- **Define the Weight Thresholds** on page 91

1.2.1. Determine the Match Fields

Before extracting data for analysis, review the types of data stored in the messages generated by each system. Use these messages to determine which fields and objects will be included in the object structure of the master index. From this object structure, select the fields to be used for matching. When selecting these fields, keep in mind how representative each field is of a specific object. For example, in a master person index, the social security number field, first and last name fields, and birth date are good representations. Certain address information or a home telephone number might also be considered. In a master company index, the match fields might include any of the fields parsed from the complete company name field, as well as a tax ID number or address and telephone information.

1.2.2. Customize the Match Configuration

Once you've determined the fields on which matching will be performed, you need to determine how the weights will be generated for each field. The primary tasks include determining whether to use probabilities or agreement weight ranges, and then choosing the best comparison functions to use for each match field.

Probabilities or Agreement Weights

The first step in configuring the match configuration is to decide whether to use m-probabilities and u-probabilities or agreement and disagreement weight ranges. Both methods will give you similar results, but agreement and disagreement weight ranges allow you to specify the precise maximum and minimum weights that can be applied to each match field, giving you control over the value of the highest and lowest matching weights that can be assigned to each record.

Defining Relative Value

For each field you will use for matching, you need to define either the m-probabilities and u-probabilities or the agreement and disagreement weight ranges in the match configuration file. Review the information provided under **"Matching Weight Formulation" on page 15** to help determine how to configure these values. Remember that a higher m-probability or agreement weight gives the field a higher weight when field values agree.

Determining the Weight Range

In order to find the initial values to set for the match and duplicate thresholds, you must determine the total range of matching weights that can be assigned to a record. This weight is the sum of all weights assigned to each match field.

Weight Ranges using Agreement Weights

For agreement and disagreement weight ranges, determining the match weight ranges that will be generated is very straightforward. Simply total the maximum agreement weights for each field to determine the maximum match weight, and total the minimum disagreement weights for each field to determine the minimum match weight. Table 41 provides a sample agreement/disagreement configuration for matching on person data. As you can see, the range of match weights generated for the master index with this configuration is from -36 to +38.

Table 41 Sample Agreement and Disagreement Weight Ranges

Field Name	**Maximum Agreement Weight**	**Minimum Disagreement Weight**
First Name	8	-8
Last Name	8	-8
Date of Birth	7	-5
Gender	5	-5
SSN	10	-10
Maximum Match Weight	38	
Minimum Match Weight		-36

Weight Ranges using Probabilities

Determining the match weight ranges when using m-probabilities and u-probabilities is a little more complicated than using agreement and disagreement weights. To determine the maximum weight that will be generated for each field, use the following formula:

$LOG_2(m\_prob/u\_prob)$

To determine the minimum match weight that will be generated for each field, use the following formula:

$LOG_2((1-m\_prob)/(1-u\_prob))$

Table 42 below illustrates a sample of m-probabilities and u-probabilities, including the corresponding agreement and disagreement weights that are generated with each combination of probabilities. As you can see, the range of match weights generated for the master index with this configuration is from -35.93 to +38

Table 42 Sample m-probabilities and u-probabilities

Field Name	**m-probability**	**u-probability**	**Max Agreement Weight**	**Min Disagreement Weight**
First Name	.996	.004	7.96	-7.96
Last Name	.996	.004	7.96	-7.96
Date of Birth	.97	.007	7.11	-5.04
Gender	.97	.03	5.01	-5.01
SSN	.999	.001	9.96	-9.96
Maximum Match Weight			38	
Minimum Match Weight				-35.93

Comparison Functions

The match configuration file defines several match types for different types of fields. You can either modify existing rows in this file, or create new rows that define custom matching logic. To determine which comparison functions to use, review the information provided in **Appendix B** **"Match Configuration Comparison Functions"**. Choose the comparison functions that best suit how you want the match fields to be processed.

1.2.3. Define the Weight Thresholds

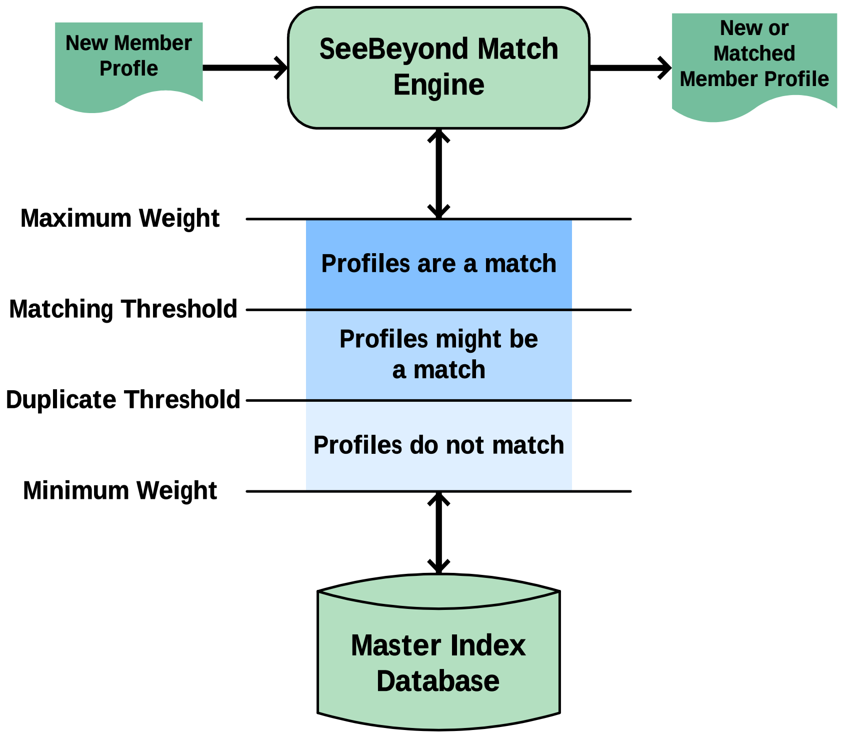
Weight Threshold Overview

Weight thresholds tell the master index how to process incoming records based on the matching probability weights generated by the SeeBeyond Match Engine. Two parameters in the Threshold configuration file provide the master index with the information needed to determine if records should be flagged as potential duplicates, if records should be automatically merged, or if a record is not a potential match to any existing records.

- **Match Threshold**—Specifies the weight at which two profiles are assumed to represent the same person and are automatically merged.
- **Duplicate Threshold**—Specifies the minimum weight at which two profiles are considered potential duplicates of one another. The matching threshold indicates the maximum weight for potential duplicates.

Figure 1 illustrates the match and duplicate thresholds in comparison to total composite match weights.

Figure 1 Weight Thresholds



Specifying the Weight Thresholds

There are two techniques for determining the initial match and duplicate thresholds. The first method, the weight distribution method, is based on the calculation of the error rates of false matches and false non-matches from analyzing the distribution spectrum of all the weighted pairs. This is the standard method, and is illustrated in Figure 2. The second method, the percentage method, relies on measuring the total maximum and minimum weights of all the matched fields and then specifying a certain percentage of these values as the initial thresholds.

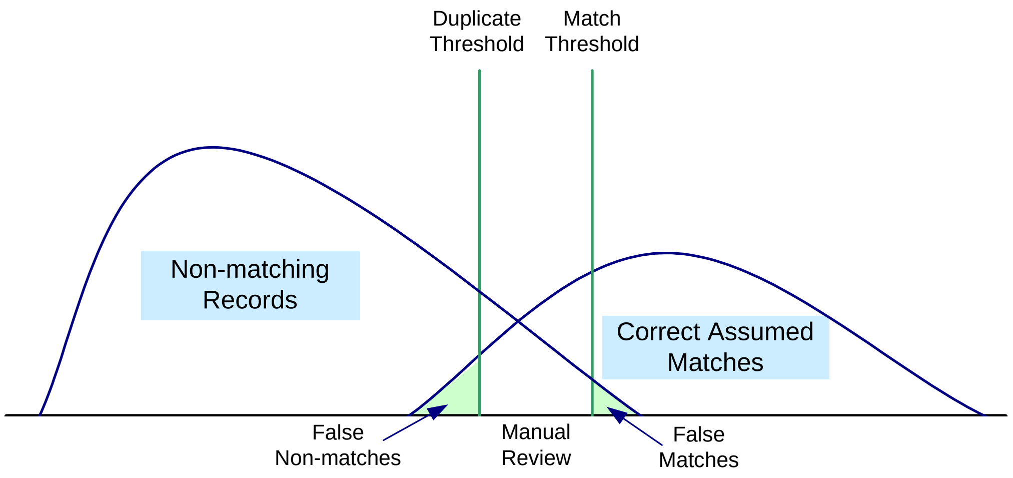
The weight distribution method is more thorough and powerful, but requires a large amount of data (match weights) to be statistically reliable. It does not apply well in cases where one candidate record is matched against very few reference records. The percentage method, though simple, is very reliable and precise when dealing with such situations. For both methods, defining the match threshold and the duplicate threshold is an iterative process.

Weight Distribution Method

Each record pair in the master index can be classified into three categories: matches, non-matches, and potential matches. In general, the distribution of records is similar to the graph shown in Figure 2. Your goal is to make sure that very few records fall into the False Matches region (if any), and that as few as possible fall into the False Non-matches region. You can see how modifying the thresholds will change this distribution. Balance this against the number of records falling within the Manual

Review section, as these will each need to be reviewed, researched, and resolved individually.

Figure 2 Weight Distribution Chart



Percentage Method

Using this method, you set the initial thresholds as a percentage of the maximum and minimum weights. Using the information provided under **“Weight Ranges using Agreement Weights”** or **“Weight Ranges using Probabilities”**, determine the maximum and minimum values that will be generated as a composite match weight. For the initial run, the the match threshold is set intentionally high to catch only the most probable matches. The duplicate threshold is set intentionally low to catch a large set of possible matches.

Set the match threshold at 70% of the maximum composite weight starting from zero as the neutral value. Using the weight range samples earlier, this would be 70% of 38, or 26.6. Set the duplicate threshold near the neutral value (that is, the value in the center of the maximum and minimum weight range). The value could be set between 10% of the maximum weight and 10% of the minimum weight. Using the samples above, this would be between 3.8 (10% of 38) and -3.6 (10% of -36).

Fine-tuning the Thresholds

Achieving the correct thresholds for your implementation is an iterative process. First, using the initial thresholds described earlier, process the data extracts into the master index database. Then analyze the assumed match and potential duplicates, paying close attention to the assumed match records with matching weights close to the match threshold, to potential duplicate records close to either threshold, and to non-matches near the duplicate threshold.

If you find that most or all of the assumed matches at the low end of the match range are not actually duplicate records, raise the match threshold accordingly. If, on the other hand, you find several potential duplicates at the high end of the duplicate range that are actual matches, decrease the match threshold accordingly. If you find that most or all of the potential duplicate records in the low end of the duplicate range should not be considered duplicate matches, consider raising the duplicate threshold. Conversely, if you find several non-matches with weight near the duplicate threshold that should be considered potential duplicates, lower the duplicate threshold.

Repeat the process of loading and analyzing data and adjusting the thresholds until you are satisfied with the results.

Appendix B

Match Configuration Comparison Functions

Match field comparison functions compare the values of a field in two records to determine whether the fields match or how closely they match. The fields are then assigned a matching weight based on the results of the comparison function. You can use several different types of comparison functions in the match configuration file in order to customize how the SeeBeyond Match Engine should match two records.

J.1 Comparison Functions

There are six primary types of comparison functions used by the SeeBeyond Match Engine. The following types of comparison functions are available.

- **Bigram Comparison Functions** on page 95
- **Uncertainty Comparison Functions** on page 96
- **Exact Comparison Function (c)** on page 97
- **Numeric Comparison Functions** on page 98
- **Date Comparison Functions** on page 100
- **Prorated Comparison Function (p)** on page 102

Certain comparison function types are very specific to the type of data being matched, such as the numeric functions and the date functions. Others, such as the Bigram and uncertainty functions, are more general and can be applied to various data fields.

Bigram Comparison Functions

The SeeBeyond Match Engine provides two different comparison functions based on the Bigram algorithm:

- Standard bigram (b1)
- Transposition bigram (b2)

A Bigram algorithm compares two strings using all combinations of two consecutive characters within each string. For example, the word "bigram" contains the following bigrams: 'bi", "ig", "gr", "ra", and "am". The Bigram comparison function returns a value between 0 and 1, which accounts for the total number of bigrams that are in common between the two strings divided by the average number of bigrams in the two strings. Bigrams handle minor typographical errors well.

Standard bigram (b1)

This is a standard Bigram comparison function, processing match fields as described above. This comparison function takes no parameters.

Transposition bigram (b2)

This comparison function is based on the standard Bigram comparison function, but handles transpositions of characters within a string. This comparison function takes no parameters.

Uncertainty Comparison Functions

The SeeBeyond Match Engine provides the following uncertainty comparison functions for comparing string fields. The first three comparison functions are generic, and the last three comparison functions are designed for specific types of information.

- Basic uncertainty (u)
- Advanced uncertainty (ua)
- Uncertainty, simplex (us)
    - Uncertainty, first name (uf)
    - Uncertainty, last name (ul)
    - Uncertainty, house number (un)

Basic uncertainty (u)

This is the standard uncertainty comparison function, and processes string fields as described above. As more differences are found between two fields, the agreement weight decreases non-linearly. Thus, the agreement weight can remain high for several differences, but will drop sharply at a certain point. This comparison function takes no parameters.

The uncertainty comparison function is based on the Jaro algorithm, which is a string comparison function that accounts for insertions, deletions, and transpositions by performing the following steps.

1. Compute the lengths of both strings to be matched.
2. Determine the number of common characters between the two strings. In order for characters to be considered common, they must be within one-half the length of the shorter string.
3. Determine the number of transpositions. A transposition means a character from the first string is out of order with the corresponding common character from the second string.

Advanced uncertainty (ua)

This comparison function is based on the standard uncertainty comparison function, **u**, with variants of Winkler/Lynch and McLaughlin. It has additional features to handle specific differences between fields, such as key punch and visual memory errors. Each feature makes use of the information made available from previous features. This comparison function takes no parameters.

These features are included in the advanced uncertainty function.

- The function determines each character in exact agreement and then assigns a value of 1.0 to each agreeing character. It then determines each disagreeing but similar character and assigns a value of 0.3 to each. Similar characters might occur because of scanning errors (for example, "1" the number versus" the letter) or keypunch errors (for example, "S" versus "D").
- The function gives increased value to agreement on the beginning characters of a string. The algorithm adjusts the weighting value up by a fixed amount if the first four characters in each string agree; it adjusts the weighting value up by smaller value if the first three, two, or one characters agree.
- The function adjusts the string comparison function value if the strings are longer than six characters and more than half of the characters after the fourth character agree.

Uncertainty, simplex (us)

This comparison function is a custom SeeBeyond version of a generic string comparison function. It is similar to the basic uncertainty comparison function, **u**, but processes data in a more simple and efficient manner, improving processing speed. The agreement weights generated by this comparison function decrease in a more uniform manner for each difference found between two fields.

Like the basic uncertainty function, the simplex function takes into account such uncertainty factors as string length, transpositions, key punch errors, and visual memory errors. Unlike the uncertainty comparison function ("u"), this function handles diacritical marks. This comparison function takes no parameters.

Uncertainty, first name (uf)

This comparison function is designed specifically for matching on first name fields, and is based on the simplex uncertainty comparison function, **us**. This comparison function analyzes the string and then adjusts the weight based on statistical data. This comparison function takes no parameters.

Uncertainty, last name (ul)

This comparison function is designed specifically for matching on last name fields, and is based on the simplex uncertainty comparison function, **us**. This comparison function analyzes the string and then adjusts the weight based on statistical data. This comparison function takes no parameters.

Uncertainty, house number (un)

This comparison function is designed specifically for matching on house numbers, and is based on the simplex uncertainty comparison function, **u**. This comparison function analyzes the string and then adjusts the weight based on statistical data. This comparison function takes no parameters.

Exact Comparison Function (c)

The SeeBeyond Match Engine provides one exact-match comparison function, "c". With this comparison function, two fields must match on each character in order to be considered a match. This comparison function takes no parameters.

Numeric Comparison Functions

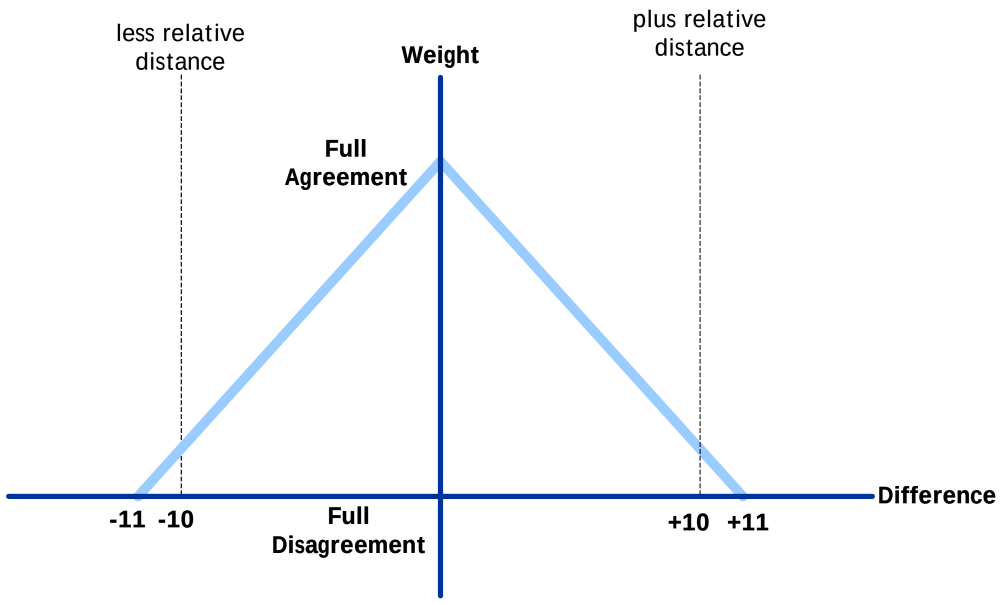
The SeeBeyond Match Engine provides several comparison functions for matching on numeric fields.

- Basic numeric (n)
- Numeric, integer (nI)
- Numeric, real (nR)
- Numeric, string (nS)

All but the nS comparison function can perform numeric string comparisons or relative distance calculations. When set for a string comparison, the functions compare numeric strings based on the advanced uncertainty comparator. When set for relative distance calculations, the matching weight between two numbers decreases as the numbers become further apart, until the relative distance plus one is reached. At this point, the numbers are considered non-matches. For example, if the relative distance is “10” and the base number for comparison is “2”, a field value of 8 receives a lower matching weight than a field value of 4; but a field value of 13 is considered a complete non-match (since the distance between 2 and 13 is 11).

Figure 3 illustrates how the weight is decreased as the difference between the two compared fields reaches the relative distance. In this diagram, the relative distance is **10** and the light blue line represents the agreement weight. When the difference between two fields reaches **11** (relative distance plus one), the fields are considered a non-match and are given the full disagreement weight.

Figure 3 Numeric Relative Distance Comparison



Basic numeric (n)

This is a basic numeric comparison function, processing numeric fields as described above. It accepts the parameters listed in Table 43.

Table 43 n, nI, and nR Comparison Function Parameters

Parameter	Description
distance-or-string	Specifies whether a relative distance calculation or a direct string comparison is used. Specify "y" to use a relative distance calculation; specify "n" to use a string comparison.
relative-distance	The greatest difference between two integers at which the values could still be considered a possible match. When the difference between two numbers is greater than the relative distance, the numbers are considered a non-match (the weight becomes zero at when the actual difference is the relative distance plus one). Only use this parameter

Numeric, integer (nI)

This numeric comparison function matches specifically on integers, and accepts the parameters listed in Table 43.

Numeric, real (nR)

This numeric comparison function matches specifically on real numbers, and accepts the parameters listed in Table 43.

Numeric, string (nS)

This numeric comparison function is designed specifically for matching on numeric strings, and is very useful for matching social security numbers. This is the only numeric comparator that can compare alphanumeric values rather than just numeric values. It accepts the parameters listed in Table 44.

Table 44 nS Comparison Function Parameters

Parameter	Description
fixed-length	An optional parameter that takes the length of the field value into account. If a fixed length is specified, the match engine considers any field of a different length to be a non-match. Specify any integer smaller than the value specified for the size specified for the field (for more information, see **"Matching Rules" on page 20**).
character-type	An indicator of whether the field must be all numeric. Specify "nu" for numeric only, or specify "an" to allow alphanumeric characters. The match engine considers any fields containing characters that are not allowed to be a non-match.

Table 44 nS Comparison Function Parameters

Parameter	Description
invalid-characters	A list of invalid characters for the field. If you specify a character, the match engine considers fields that consist of only that character to be a non-match. For example, if you specify "0", then an SSN field cannot contain all zeros. Specify as many alphanumeric characters as needed, separated by a space.

Date Comparison Functions

The SeeBeyond Match Engine provides various date comparison functions. When comparing dates, the match engine compares each date component (for example, it compares the year in the first date against the year in the second date, the month against the month, and the day against the day). This allows for multiple transpositions in each date field. The date comparators use the Java date format (java.sql.Date), allowing the comparator to use the Gregorian calendar and to take into account the time zone where the date field originated.

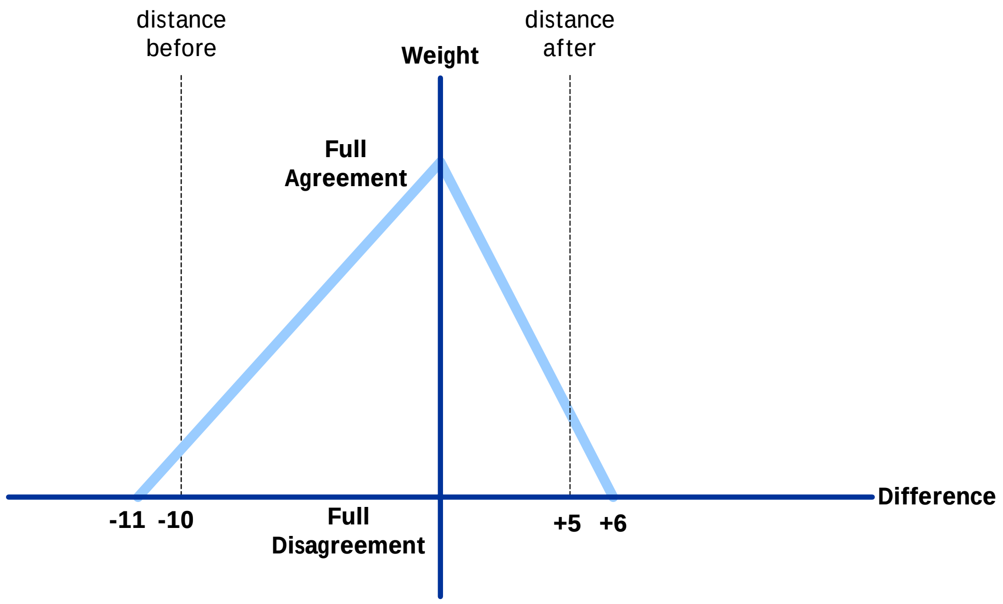
The following comparison functions are available for matching on date fields.

- Date, year (dY)
- Date, month (dM)
- Date, day (dD)
- Date, hour (dH)
- Date, minute (dm)
- Date, second (ds)

As with the numeric comparison functions, the date comparison functions can use either a direct string comparison or a relative distance calculation. When using a relative distance calculation, the matching weight between two dates decreases as the dates become further apart, until the relative distance is reached. When the difference becomes the relative distance plus one, the dates are considered non-matches. You can specify different relative distances for before and after the given date. Any dates falling outside of the specified time period receive a complete disagreement weight. The relative distances are specified in the smallest unit of time being matched (this differs for each date comparison function).

Figure 4 illustrates how the weight is decreased as the difference between the two compared fields reaches either the before or after relative distance. In this diagram, the before relative distance is **11**, the after relative distance is **5**, and the light blue line represents the agreement weight. When the base date is later than the compared date and the difference between the dates reaches **11** (distance before plus one), the fields are considered a non-match and are given the full disagreement weight. When the base date is earlier than the compared date and the difference between the dates reaches **6** (distance after plus 1), the fields are considered a non-match.

Figure 4 Date Relative Distance Comparison



The date comparison functions take the parameters listed in Table 45.

Table 45 Date Comparison Function Parameters

Parameter	Description
distance-or-string	Specifies whether a relative distance calculation or a direct string comparison is used. Specify “y” to use a relative distance calculation; specify “n” to use a string comparison.
distance-before	The number of units prior to the reference date/time for which two date fields can still be considered a match.
distance-after	The number of units following the reference date/ time for which two date fields can still be considered a match.

Date, year (dY)

This date comparison function takes only the 4-character year into account for matching. If relative distance calculation is specified, the relative distance is specified in years.

Date, month (dM)

This date comparison function takes the month and year into account for matching. If relative distance calculation is specified, the relative distance is specified in months.

Date, day (dD)

This date comparison function takes the day, month, and year into account for matching. If relative distance calculation is specified, the relative distance is specified in days.

Date, hour (dH)

This date comparison function takes the hour, day, month, and year into account for matching. If relative distance calculation is specified, the relative distance is specified in hours.

Date, minute (dm)

This date comparison function takes the minute, hour, day, month, and year into account for matching. If relative distance calculation is specified, the relative distance is specified in minutes.

Date, second (ds)

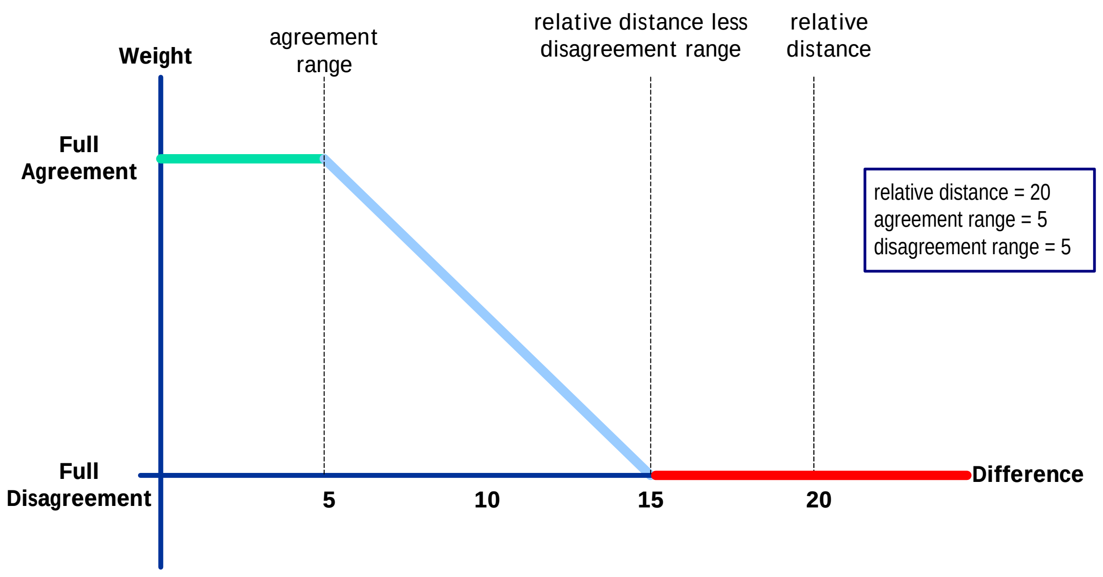
This date comparison function takes the second, minute, hour, day, month, and year into account for matching. If relative distance calculation is specified, the relative distance is specified in seconds.

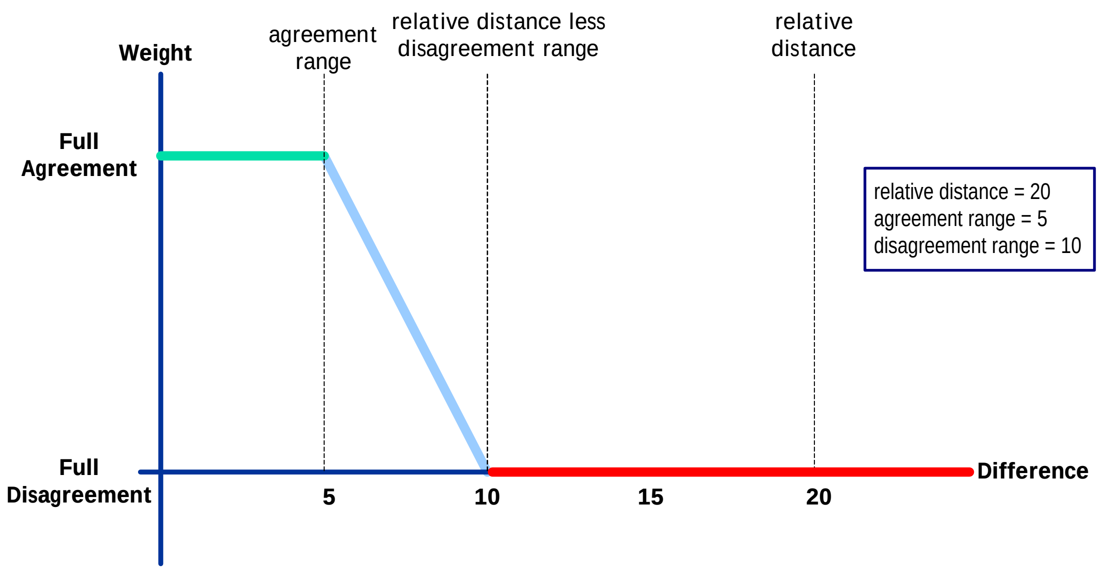
Prorated Comparison Function (p)

The prorated comparison function uses a relative distance calculation and allows you to specify how quickly the agreement weight between two fields decreases. Matching weights are assigned with a linear adjustment according to the parameters you specify. You specify an initial agreement range. If the difference between two fields falls within that range, the fields are considered a complete match. You also specify a disagreement range ending with the relative distance. If the difference between two fields falls within that range, the fields are considered a non-match. When the difference between the fields falls between those two ranges, they are considered to be partial matches and the agreement weight is adjusted linearly. Any difference greater than the relative distance is always considered a non-match.

Figure 5 illustrates how weighting is adjusted per the parameters you define. In these diagrams, the green line indicates full agreement, the light blue line indicates prorated agreement, and the red line indicates full disagreement. The diagrams illustrate how increasing the disagreement weight causes the prorated agreement weight to decrease more sharply.

Figure 5 Prorated Linear Adjustment Comparison





The prorated comparison functions takes the parameters listed in Table 46.

Table 46 Prorated Comparison Function Parameters

Parameter	Description
relative-distance	The greatest difference between two numbers at which they can still be considered a match or partial match.
agreement-range	The greatest difference between two numbers at which they are considered a full match. This number must be less than the relative distance.

Table 46 Prorated Comparison Function Parameters

Parameter	Description
disagreement-range	This number indicates the minimum difference at which two numbers are considered a non-match and shortens or lengthens the weighting scale. To find this difference, the match engine subtracts this value from the relative distance. If the fields differ by that amount or greater, they are considered to be a non-match. The weighting scale decreases in size as the value of the full-disagreement parameter increases (see diagram).

J.2 Comparison Function Options

The options listed below can be used in conjunction with the above string comparison function to give them more functionality. For example, you can use an 'ufI' string comparison function that refers to first name comparison function with the possibility to switch fields if the first one disagree

- **I**—This is a major inversion option that allows for field transpositions. If two compared fields do not match, this option lets the match engine know to compare the original field in the first record with the next field in the second record. If those fields agree, the match engine assigns the full agreement weight and switches the fields.
- **i**—This is a minor inversion option similar to the major inversion (I) described above. This option only assigns one-half of the full agreement weight if the transposed fields match.
- **x**—If two or more fields with this option match, their weight is doubled; but if any of the fields with this option disagree, the weight is not doubled.
- **k**—This option can be used with the 'x' option to give more importance to one field. Specifying this option on a field tells the match engine to double the match weight for a sub-group of fields with the 'x' option by doubling the weight as soon as it comes to the field with the 'k' option.

Glossary

agreement weight
A positive weight assigned to a match field if the values agree between two fields.

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.

comparison function
A command specific to the SeeBeyond Match Engine that specifies how two fields are compared. Comparison functions are specified for each match field in the match configuration file.

disagreement weight
A negative weight assigned to a match field if the field values disagree between two fields.

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 file that tells the SeeBeyond Match Engine which matching rules in the match configuration file to use for weighting.

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*.

standardization type
An indicator specified in the **StandardizationConfig** section of the Match Field file that tells the SeeBeyond Match Engine how to standardize information.

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