Oracle® Healthcare Master Person Index

Configuration Reference

Release 1.1

E18592-01

February 2011

ORACLE®

Oracle Healthcare Master Person Index Configuration Reference, Release 1.1

E18592-01

Contents

3   Master Person Index Field Notations

Preface

The *Oracle Healthcare Master Person Index Configuration Reference* provides conceptual information for configuring the Oracle Healthcare Master Person Index (OHMPI) and the Master Index Data Manager (MIDM).

Audience

This document is intended for users of the Oracle Healthcare Master Person Index and/or the Master Index Data Manager who need to set up or change their system's configuration. If is a companion piece to the *Oracle Healthcare Master Person Index Configuration Guide* (Part Number E18473-01) and should be used in conjunction with it.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documents

For more information and instructions for implementing and using a master index application, see the following documents in the Oracle Healthcare Master Person Index Release 1.1 documentation set:

- *Oracle Healthcare Master Person Index Installation Guide*

- *Oracle Healthcare Master Person Index Release Notes*

- *Oracle Healthcare Master Person Index User's Guide*

- *Oracle Healthcare Master Person Index Data Manager's Guide*

- *Oracle Healthcare Master Person Index Data Manager's Guide*

- *Oracle Healthcare Master Person Index Configuration Guide*

- *Oracle Healthcare Master Person IndexMatch Engine Reference*

- *Oracle Healthcare Master Person Index Standardization Reference*

- *Oracle Healthcare Master Person Index Analyzing and Cleansing Data User's Guide*

- *Oracle Healthcare Master Person Index Loading the Initial Data Set User's Guide*

- *Oracle Healthcare Master Person Index Command Line Reports and Database Maintenance User's Guide*

- *Oracle Healthcare Master Person Index Working With IHE Profiles*

- *Oracle Healthcare Master Person Index WebLogic User's Guide*

> **Note:** These documents are designed to be used together when implementing a master index application.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction to the Master Person Index Configuration Reference

This chapter provides an introduction to the Oracle Healthcare Master Person Index (OHMPI), including its flexible framework, features, and configuration.

This chapter includes the following section:

- "Introducing the Flexible Framework of Oracle Healthcare Master Person Index"

Introducing the Flexible Framework of Oracle Healthcare Master Person Index

Oracle Healthcare Master Person Index provides a flexible framework that allows you to create matching and indexing applications called **enterprise-wide master person index applications**. It is an application building tool to help you design, configure, and create a master person index application that will uniquely identify and cross-reference the business objects stored in your system databases. Business objects can be any type of entity for which you store information, such as customers, patients, vendors, businesses, inventory, and so on.

The following topics provide additional information about Oracle Healthcare Master Person Index:

- "Oracle Healthcare Master Person Index Configuration"
- "Oracle Healthcare Master Person Index Features"

Oracle Healthcare Master Person Index Configuration

In Oracle Healthcare Master Person Index, you define the data structure of the business objects to be stored and cross-referenced. In addition, you define the logic that determines how data is updated, standardized, weighted, and matched in the master person index database. The structure and logic you define is located in a group of XML configuration files that you create using the wizard. These files are created within the context of a NetBeans project, and can be further customized using either the Master Person Index Configuration Editor or the NetBeans XML editor. This document describes the structure of the XML files and how each configuration option affects the master person index application.

Oracle Healthcare Master Person Index Features

Oracle Healthcare Master Person Index provides features and functions to allow you to create and configure a master person index application for any type of data. The primary function of Oracle Healthcare Master Person Index is to automate the creation

of a highly configurable master person index application. A wizard guides you through the initial setup steps, and the Master Person Index Configuration Editor allows you to further customize the configuration of the master person index application. The components you need to implement a master person index application are automatically generated.

Oracle Healthcare Master Person Index provides the following features:

- **Rapid Development** - Rapid and intuitive development of a master person index application using a wizard to create the master person index configuration and using XML documents to configure the attributes of the index. Templates are provided for quick development of person and company object structures.

- **Automated Component Generation** - Oracle Healthcare Master Person Index automatically creates the configuration files that define the primary attributes of the master person index application, including the configuration of the Master Index Data Manager (MIDM). Oracle Healthcare Master Person Index also generates scripts that create the appropriate database schemas.

- **Configurable Survivor Calculator** - Oracle Healthcare Master Person Index provides predefined strategies for determining which field values to populate in the single best record (SBR). You can define different survivor rules for each field, and you can create a custom survivor strategy to implement in the master person index application.

- **Flexible Architecture** - Oracle Healthcare Master Person Index provides a flexible platform that allows you to create a master person index application for any business object. You can customize the object structure so the master person index application can match and store any type of data, allowing you to design an application that specifically meets your data processing needs.

- **Configurable Matching Algorithm** - Oracle Healthcare Master Person Index provides standard support for the Oracle Healthcare Master Person Index Match Engine (OHMPI Match Engine). In addition, you can plug in a custom matching algorithm to the master person index application.

- **Custom Java API** - Oracle Healthcare Master Person Index generates a Java API that is customized to the object structure you define.

- **Standard Reports** - Oracle Healthcare Master Person Index provides a set of standard reports with each master person index application that can be run from a command line or from the MIDM. The reports help you monitor the state of the data stored in the master person index application and help you identify configuration changes that might be required.

2

[[Oracle Healthcare Master Person Index Configuration]

[This chapter provides an overview of the configurable components of an Oracle Healthcare Master Person Index (OHMPI).]

[This chapter includes the following sections:]

- [[[["Introducing the Oracle Healthcare Master Person Index Editor and Configuration Files"]]]

- [[["Configuring Master Person Index Object Definitions"]]]

- [[["Configuring Queries"]]]

- [[["Configuring Manager Service Components"]]]

- [[["Configuring Match Fields"]]]

- [[["Configuring Survivor Strategy"]]]

- [[["Configuring SBR, Matching, and Blocking Filters"]]]

- [[["Configuring Field Validation"]]]

- [[["Configuring Master Index Data Manager"]]

  [The above sections are reference only. For instructions on configuring a master person index application, see [*Oracle Healthcare Master Person Index Configuration Guide*].]]]

[[Introducing the Oracle Healthcare Master Person Index Editor and Configuration Files]

[The files that configure the components of the master person index application are created by the wizard and define characteristics of the application, such as how data is processed, queried, and matched, and how it appears on the Master Index Data Manager (MIDM). These files configure the runtime components of the master person index application.]

[The following sections provide an overview of the configurable components of a master person index application and of the configuration files that define processing properties and the data structure of the master person index application. They also describe the relationships between these files.]

- [[[["Understanding Oracle Healthcare Master Person Index Configuration Files"]]]

- [[["Using the Editors for Oracle Healthcare Master Person Index"]]]]

[[Understanding Oracle Healthcare Master Person Index Configuration Files]

[Several XML configuration files define primary characteristics of the master person index application, such as how data is processed, queried, and matched. These files configure runtime components of the master person index application.]

[The configuration files include the following:]

- [[[["Master Person Index object.xml File"]]]
- [[[["Master Person Index query.xml File"]]]
- [[[["Master Person Index mefa.xml File"]]]
- [[[["Master Person Index master.xml File"]]]
- [[[["Master Person Index update.xml File"]]]
- [[[["Master Person Index filter.xml"]]]
- [[[["Master Person Index validation.xml File"]]]
- [[[["Master Person Index security.xml File"]]]
- [[[["Master Person Index edm.xml File"]]]
- [[[["Match and Standardization Engine Configuration Files"]]]]

[[Master Person Index object.xml File]

[In the wizard, you define the objects and fields contained in the object structure, along with properties for those fields. The information you specify is written to object.xml in the master person index project. This file defines the objects stored in the master person index application and their relationships to one another. It also defines the fields contained in each object, as well as certain properties of each field, such as length, data type, whether it is required, whether it is a unique key, and so on. This file contains one parent object; all other objects must be child objects to that parent object. The object structure you define in object.xml determines the structure of the database tables that store object data and the structure of the Java API.]]

[[Master Person Index query.xml File]

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

[[Master Person Index mefa.xml File]

[In mefa.xml, you configure the [**Matching Service**] by specifying the fields to be standardized and the fields to be used for matching, as well as defining how the fields are standardized and matched. It also specifies the match and standardization engines to use and the query process for matching. Standardization includes defining fields to be reformatted (or parsed), normalized, or converted to their phonetic version. For matching, you must also define the data string to be passed to the match engine. The rules you define for standardization and matching are dependent on the match and standardization engines in use. [*Oracle Healthcare Master Person Index Match Engine*

Reference] (Part Number E18470-01) and [*Oracle Healthcare Master Person Index Standardization Engine Reference*] (Part Number E18471-01) describe the rules for the OHMPI Engine and OHMPI Standardization Engine.]

[In addition, master.xml, described below, also configures the match process by defining certain match parameters that define weight thresholds, how assumed matches are processed, and how potential duplicates are processed. It also specifies the query to use for matching.]]

[[Master Person Index master.xml File]

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

[master.xml also configures the EUIDs assigned by the master person index application. You can specify an EUID length, whether a checksum value is used for additional verification, and a "chunk size". Specifying a chunk size allows the EUID generator to obtain a block of EUIDs from the sbyn_seq_table database table so it does not need to query the table each time it generates a new EUID.]]

[[Master Person Index update.xml File]

[In update.xml, you can define formulas that determine which data in an enterprise record should be considered the most reliable and how updates to the single best record (SBR) will be handled. The survivor calculator uses these formulas to decide what data from each system record to include in each object's SBR. The SBR is the portion of the enterprise record that represents the data that is considered to be the most accurate and current for an object.]

[The SBR is defined by a mapping of fields from external system records. Since there might be many external systems, you can optionally specify a strategy to select the value for an SBR field from the list of external values. You can also specify any additional fields that might be required by the selection strategy to determine which external system contains the best data, such as the object's update date and time.]

[This file also allows you to specify custom update procedures that you define in custom Java code you can plug in to the application. You can create Java classes that define special processing to perform against a record when the record is created, updated, merged, or unmerged. These classes must be created in the Source Packages folder of the EJB project and can be specified for each transaction type in update.xml.]]

[[Master Person Index filter.xml]

[You can further configure the survivor calculator, blocking query, and match process by defining exclusion lists in filter.xml. Exclusion lists allow you to define values that should not be populated into the SBR, that should not be considered in the composite matching weight, and that should be ignored in the blocking query. Values you would want to filter out primarily include default values that are used when the actual value for a field is unknown. Default values can cause the blocking query to return records that are not a close match and can skew matching results.]]

[[Master Person Index validation.xml File]

[By default, validation.xml (validation.xml) defines certain validations for the local identifiers assigned by each external system. You can create custom Java classes that

define rules for validating field values before they are saved to the master person index database. You can then specify the Java classes in validation.xml to make them part of the Oracle Healthcare Master Person Index application.]]

[[Master Person Index security.xml File]

[This file defines security roles and permissions for the client applications that access the master person index database.]]

[[Master Person Index edm.xml File]

[Configuration of the appearance and certain processing properties of the MIDM is contained in midm.xml. In this file, you define each object and field that appears on the MIDM, along with the properties of each field, such as the field type and length, field labels, format masks, and so on. You can also define the order in which objects and fields appear on the MIDM pages.]

[This file defines several additional properties of the MIDM, including the types of searches available, whether wildcard characters can be used, the criteria for the searches, and the results fields that appear. You can also specify whether an audit log is maintained of each instance data is accessed through the MIDM. For healthcare-based master person index applications, this supports the privacy rules mandated by the HIPAA regulation for healthcare. This file also includes the configuration of the reports generated from the MIDM.]

[Finally, midm.xml defines certain implementation information, such as the application server in use, debugging rules, and security activation.]

[The files that configure the components of the master person index application are created by the wizard and define characteristics of the application, such as how data is processed, queried, and matched, and how it appears on the Master Index Data Manager (MIDM). These files configure the runtime components of the master person index application.]]

[[Match and Standardization Engine Configuration Files]

[Several match and standardization engine configuration files are included in the project tree. You can customize matching logic and standardization information for the match and standardization engines by modifying these files. The match configuration file, which defines and configures the comparator functions, can be modified using the Master Person Index Configuration Editor or the NetBeans text editor. The standardization files, which provide information to the standardization engine about how data should be parsed and normalized, can be modified using the text editor.]

[For information about the structure of these files and how they can be modified, see [*Oracle Healthcare Master Person Index Match Engine Reference*] (Part Number E18470-01) and [*Oracle Healthcare Master Person Index Standardization Engine Reference*] (Part Number E18471-01).]]]

[[Using the Editors for Oracle Healthcare Master Person Index]

[You can use the NetBeans XML editor or the Master Person Index Configuration Editor to modify the configuration files created by the wizard. The Configuration Editor provides a series of windows to help guide you though the configuration of master person index application components. The NetBeans XML editor allows you to modify the XML code directly.]

[The following sections provide additional information about the editors:]

- [[[["XML Editors"]]]]

- [[["Master Person Index Configuration Editor"]]]]

[[XML Editors]

[If you are familiar with XML, you can configure the master person index applications by modifying the XML code directly. Use caution when modifying the XML files because there are dependencies between files. For example, all fields listed in any of the configuration files must also be defined in object.xml. Any queries referenced in midm.xml must also be defined in the query.xml.]]

[[Master Person Index Configuration Editor]

[The Master Person Index Configuration Editor allows you to modify most, but not all, configuration elements for a master person index application using a graphical user interface. You can also use the editor to modify the match configuration file for the OHMPI Match Engine, but not to modify the standardization configuration files. While you can use the Configuration Editor to modify most of the configuration files, some elements can only be modified using the NetBeans XML editor. Following is a summary of which features can be configured using the Configuration Editor and which need to be modified using the XML editor.]

[The object.xml File]

[You can modify most elements of object.xml using the Configuration Editor. The following can only be modified using the XML editor:]

- [[[Database type]]
- [[Date format]]
- [[Maximum field value]]
- [[Minimum field value]]]

[It is not recommended that you change the database type, but if you modify the database type or date format elements, you need to regenerate the application to create the updated database scripts. This does not recreate the Systems or Code Lists scripts; you need to update those manually.]

[query.xml]

[You can modify all elements in query.xml using the Configuration Editor. If you create a query to use in the Master Index Data Manager (MIDM) or to use for the matching query, you need to add the query to the appropriate file (master.xml or midm.xml) manually.]

[master.xml]

[Most elements in master.xml cannot be modified using the Configuration Editor. You can modify the duplicate and match thresholds from the Configuration Editor.]

[mefa.xml]

[You can use the Configuration Editor to modify all commonly modified elements in mefa.xml, including defining standardization structures, normalization structures, and phonetic encoding. If you create custom classes to implement a block picker, pass controller, match engine, or standardization engine, you need to specify the implementation classes in this file using the XML editor.]

[update.xml]

[The Configuration Editor does not modify update.xml. If you make any changes to the object structure, review this file to verify that all fields or objects are included in the survivor strategy and that the field and object names are correct.]

[update.xml]

[The Configuration Editor does not modify validation.xml. If you create a custom field validation class, you need to specify the implementation class in this file using the XML editor.]

[edm.xml]

[Several elements in midm.xml are not modified using the Configuration Editor. You can add and delete fields that appear on the MIDM and modify the display name and the value and input masks. All other field properties can only be modified using the XML editor.]

[Field integrity is maintained when you delete a field using the Configuration Editor. The field is automatically deleted from the MIDM object structure and from any MIDM page definitions that include the field, such as a search page or report.]

[Match Configuration File]

[You can modify all components of the Match Configuration file using the Configuration Editor, including adding and removing comparators. The Configuration Editor does not validate the extra parameters that can be used for certain comparators, so you should verify your changes by reviewing the match configuration file manually.]]]]

[[Configuring Master Person Index Object Definitions]

[The properties for the objects you will store in the master person index database are defined in object.xml. This file defines the parent and child objects to be indexed and the fields contained in each object, including key properties for each field, such as the field size, unique record identifiers, and whether certain fields are required or can be updated. After you define the master person index framework and create the configuration files, you can modify the object structure that you defined.]

[The Object Definition is used as a basis for most of the master person index application components. The information you specify for this file defines the dynamic Java API and the database structure for the primary tables that store object information in the master person index application.]

[The following sections describe object.xml, which defines the object structure.]

- [[[["Master Person Index Object Definition Components"]]]]
- [[["The Master Person Index object.xml File"]]]]

[[Master Person Index Object Definition Components]

[The object definition includes three primary components that together define the structure of the data in the master person index application. Most configuration files in the master person index application rely on the objects and fields defined in the Object Definition. For example, the fields you specify for the match string, queries, standardization, and the survivor calculator must all be defined in the Object Definition.]

[The following sections describe each component of the object definition:]

- [[[["Master Person Index Object Definition Objects"]]]
- [[["Master Person Index Object Definition Fields"]]]
- [[["Master Person Index Object Definition Relationships"]]]]

[[Master Person Index Object Definition Objects]

[In a master person index application, information is stored in objects. Each object in the data structure represents a different type of information. For example, if you are indexing businesses, you might have one object type to store general information about the business (such as the business name and type), one to store address information, and one to store contact information. When indexing personal information, you might have one object type to store general information about the person (such as their name, date of birth, and gender), one to store address information, and one to store telephone information. The object structure can have several objects, but only one primary object (called the parent object). This object is the parent to all other objects defined in the Object Definition. The object structure can have multiple child objects or no child objects at all.]

[Generally, a record in the master person index application has information in one parent object and multiple child objects. A record can also have multiple instances of each child object. For example, in the person index example above, a record for a single person would have one name, one date of birth, and one gender, all three stored in the parent object. However, the same record might have several different addresses, each of which is stored in a separate Address object.]]

[[Master Person Index Object Definition Fields]

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

[[Master Person Index Object Definition Relationships]

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

[[The Master Person Index object.xml File]

[The object structure is defined in object.xml. The information entered into the default configuration file is based on the objects and fields you defined in the wizard. Depending on how completely you defined the object structure in the wizard, this file should not require customization.]

[The following sections provide information about working with object.xml:]

- [[[["Modifying the Master Person Index Object Definition"]]]
- [[["The object.xml File Structure"]]]]

[[Modifying the Master Person Index Object Definition]

[When you use the wizard to define the object structure, all the configuration files for the master person index application are automatically generated based on the information you provide. You can modify object.xml at any time prior to deploying the

associated project, but you must regenerate the application and redeploy the project after doing so. If you modify the object structure using the configuration editor, the remaining configuration files are updated accordingly to keep them synchronized. If you update object structure by modifying the file directly, you also need to update the remaining configuration files. For example, if you modify the file directly and you delete a field from the object structure that also appears on the MIDM, appears in the SBR, and is defined for standardization and matching, you must remove the field from midm.xml, update.xml, and mefa.xml. Any changes made to the file without regenerating the project will not take effect.]

[The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes.]]

[[The object.xml File Structure]

[This section describes the structure of the XML file, general requirements, and constraints. It also provides a sample implementation.]

[object.xml File Description]

[["Table 1 object.xml File Structure"] lists each element in object.xml and provides a description of each element along with any requirements or constraints for each element.]

[Table 1 object.xml File Structure]

[Element/Attribute]	[Description]
[name]	[The name of the master person index application. This name must match the name of the parent object.]
[database]	[The database platform used by the master person index application. Specify [**MySQL**], [**Oracle**], or [**SQL Server**].]
[dateformat]	[The date format to use for the master person index application. Three formats are allowed for the date: MM/dd/yyyy, yyyy/MM/dd, and dd/MM/yyyy.]
[nodes]	[The configuration information for an object. There can be multiple nodes elements, each defining one parent or child object in the object structure. Each nodes element also defines the fields contained in each object along with the field attributes. The object structure must include one parent object and can include several child objects or no child objects.]
[tag]	[The name of the parent or child object defined by the nodes element.] [Note: Due to database naming constraints, the length of the name of the parent object plus the length of any child object name must be 21 characters or less.]
[fields]	[The configuration information for a field. There can be multiple fields elements.]
[field-name]	[The name of the field. Follow these guidelines when naming fields.] ▪ [[[The name cannot be longer than 30 characters.]] ▪ [[The name cannot be objectId, where object is the name of an object in the data structure. For example, you cannot create a field named AddressId if there is an Address object in the structure.]] ▪ [[Field names must conform to naming conventions for the database you are using, must conform to Java naming standards, and cannot contain XML reserved characters.]]]

[Element/Attribute]	[Description]
[field-type]	[The data type for each field. Possible data types are:] ■ [[[[**string**] - Fields of this type contain a string of characters.]] ■ [[[**date**] - Fields of this type contain a date value.]] ■ [[[**float**] - Fields of this type contain a floating point integer.]] ■ [[[**int**] - Fields of this type contain an integer.]] ■ [[[**byte**] - Fields of this type contain a single character.]] ■ [[[**boolean**] - Fields of this type can contain either [**true**] or [**false**].]]]
[size]	[The number of characters allowed in each field. If you modify this value, be sure to modify the corresponding database column accordingly.]
[updateable]	[An indicator of whether the field can be updated using the MIDM or from back-end messages. Specify [**true**] if the field can be updated, or specify [**false**] if it cannot.]
[required]	[An indicator of whether the field is required in order to save an enterprise object in the database. Specify [**true**] if the field is required, or specify [**false**] if it is not.]
[code-module]	[The identification code for the menu list that appears for this field in the MIDM. This must match a value in the code column of the sbyn_ common_header database table. This element is optional.]
[maximum-value]	[The maximum value allowed in the field. To specify a value for a date field, use the format YYYY-MM-DD. This element is optional.]
[minimum-value]	[The minimum value allowed in the field. To specify a value for a date field, use the format YYYY-MM-DD. This element is optional.]
[pattern]	[The required pattern for the field. For more information about possible values and using Java patterns, see "Patterns" in the class list for java.util.regex in the Javadocs provided with J2SE Platform. This element is optional.]
[user-code]	[The processing code for the drop-down list that appears on the MIDM for the fields defined by the constraint-by property, described below. These codes are used for non-unique IDs, such as account numbers, insurance policies, credit cards, and so on.] [Note: This must match an entry in the code_list column of the sbyn_ user_code database table.]
[constraint-by]	[The name of the field that contains the corresponding user-code value (described above) to use to validate the current field. The user-code and constraint-by properties are used in conjunction to define non-unique ID types, such as credit card numbers or account numbers. The first purpose is to define a drop-down list for the field that contains the user code value. The second purpose is to validate the field that contains the constraint value against definitions for the field with the user code value.] [For example, if you store non-unique IDs such as credit card numbers or insurance policy numbers, you could create a field named [**ID Type**] with a user-code of CREDCARD (CREDCARD also needs to be defined as a code in the sbyn_user_code table). This gives the [**ID Type**] field a drop-down list based on the definitions for CREDCARD in the sbyn_ user_code table. Definitions would be VISA, MASTERCARD, AMEX, and so on. You could then create a field named [**ID**] that would be constrained by the formats defined for the [**ID Type**] field. Any credit card numbers you enter would be validated against the format defined for the type of credit card you selected in [**ID Type**].]

[Element/Attribute]	[Description]
[key-type]	[An indicator of whether the field is used to identify unique objects. Specify [**true**] if the element is a unique object identifier; specify [**false**] if it is not. This element is optional.]
	[Note: Each child object should contain at least one field that is a unique object identifier, but it is not required. If two or more fields are unique identifiers, the combined value of these fields must be unique in a given enterprise record.]
[relationships]	[The configuration information for the hierarchy of the objects you defined in the nodes elements. Only one object can be the parent object; the remaining objects must be defined as children. The relationship definition allows a record to contain multiple instances of each child object. For example, if you define Address and Telephone child objects, the record can contain multiple addresses and telephone numbers.]
[name]	[The name of the parent object, as defined in the nodes elements.]
[children]	[The name of a child object, as defined in the nodes elements. You can define multiple children elements.]

[object.xml Example]

[Following is a short sample illustrating the elements in object.xml. The DOB field shows usage of the minimum-value element, the SSN field shows usage of the pattern element, and the AddressType field illustrates the code-module element. The AddressType field also has the key-type set to true, meaning that each record can only contain one address of each address type.]

```
[[<name>Person</name>]
[    <database>oracle</database>]
[    <dateformat>MM/dd/yyyy</dateformat>]
[    <nodes>]
[        <tag>Person</tag>]
[        <fields>]
[            <field-name>LastName</field-name>]
[            <field-type>string</field-type>]
[            <size>40</size>]
[            <updateable>true</updateable>]
[            <required>true</required>]
[            <key-type>false</key-type>]
[        </fields>]
[        <fields>]
[            <field-name>FirstName</field-name>]
[            <field-type>string</field-type>]
[            <size>40</size>]
[            <updateable>true</updateable>]
[            <required>true</required>]
[            <key-type>false</key-type>]
[        </fields>]
[        <fields>]
[            <field-name>DOB</field-name>]
[            <field-type>date</field-type>]
[            <updateable>true</updateable>]
[            <required>true</required>]
[            <minimum-value>1900-01-01</minimum-value>]
[            <key-type>false</key-type>]
[        </fields>]
[        <fields>]
[            <field-name>SSN</field-name>]
[            <field-type>string</field-type>]
```

```
[          <size>16</size>]
[          <updateable>true</updateable>]
[          <required>false</required>]
[          <pattern>[0-9]{9}</pattern>]
[          <key-type>false</key-type>]
[       </fields>]
[    </nodes>]
[    <nodes>]
[       <tag>Address</tag>]
[       <fields>]
[          <field-name>AddressType</field-name>]
[          <field-type>string</field-type>]
[          <size>8</size>]
[          <updateable>true</updateable>]
[          <required>true</required>]
[          <code-module>ADDRTYPE</code-module>]
[          <key-type>true</key-type>]
[       </fields>]
[       ...]
[    </nodes>]
[    <nodes>]
[       <tag>Phone</tag>]
[       ...]
[    </nodes>]
[    <relationships>]
[       <name>Person</name>]
[       <children>Address</children>]
[       <children>Phone</children>]
[    </relationships>]
[]]]]]
```

[[Configuring Queries]

[In query.xml, you configure properties of the Query Builder, which is a class that uses defined criteria and options to generate queries and query results from a master person index database. The criteria and options used by the Query Builder to create database queries are defined in query.xml. The criteria must be fields that are defined in the Object Definition, and the options are key and value pairs that fine-tune the query operation. You can define the characteristics of the searches performed from the Master Index Data Manager and of the queries used by the master person index application to search for a candidate pool of potential matches for incoming records.]

[The following sections provide information about queries and the structure of query.xml:]

- [[[["Query Builder Components"]]]]

- [[[["Range Searching"]]]]

- [[[["The query.xml File"]]]]

- [[[["Range Search Processing"]]]]

[[Query Builder Components]

[The master person index application performs two types of queries. Users perform manual queries from the MIDM and the master person index application automatically performs queries before processing matches for an incoming record. Two types of queries, [**basic queries**] and [**blocking queries**], are predefined in the Query Builder. By default, basic queries are defined for the MIDM and blocking

queries are defined for match processing, though this is not required. You can also use a blocking query for the phonetic searches performed from the MIDM. Both types of queries are configured by query.xml, and custom queries can be created and implemented with the master person index application.]

[You can configure certain query properties. You can configure both basic and blocking queries to search on standardized or phonetic versions of the search criteria, and you can also specify that they search on exact values or a range of values. Basic queries can be configured to allow wildcard characters. For the blocking queries, you define the criteria to include in each block of query criteria.]

[The following topics provide additional information about the different types of queries:]

- [[[["Basic Queries in a Master Person Index"]]]]
- [[["Blocking Queries in a Master Person Index"]]]]
- [[["Phonetic Queries in a Master Person Index"]]]]]

[[Basic Queries in a Master Person Index]

[By default, searches performed from the MIDM follow the logic defined in the configured basic queries. You can specify which query type to use for each search defined for the MIDM (this is specified in midm.xml). These searches can be weighted, which means that the match engine calculates the likelihood that the search results match the actual search criteria and assigns a matching weight to each returned record. You can specify whether the search is performed on the original or phonetic version of the criteria.]

[The basic query uses all supplied search criteria to create a single SQL query. For this query, each field in the WHERE clause is joined by an AND operator, meaning that only records that match on all search fields are returned. This query has an option to allow wildcard characters in the search criteria (a percent sign (%) indicates multiple unknown characters). When this option is set to [**true**], the query uses the LIKE operator rather than EQUALS. This option allows you to search by criteria for which you have incomplete data.]

[The searches performed from the MIDM can be further customized in midm.xml (for more information, see ["Configuring Master Index Data Manager"]).]]

[[Blocking Queries in a Master Person Index]

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

[Blocking Query Block Processing]

[In query.xml, you define the criteria and conditions for querying the database to retrieve the subset of possible matches to the incoming record, including hints. You can define multiple queries, known as [**blocks**], for each blocking query, and the master person index application performs each of these queries in turn until sufficient records are retrieved (called a [**match pass**]). Using the default Query Builder, a block is only processed if the search criteria include all of the fields defined for that block. Each field in a block is joined by an AND operator in the WHERE clause, and each

block is joined by a UNION operator. This type of search can also be used as a phonetic search in the MIDM.]

[Blocking Query for Matching]

[The blocking queries you define here are referenced in master.xml, which specifies which one of the defined blocking queries to use for match processing. They might also be referenced in midm.xml if a blocking query is used for phonetic searches from the MIDM. To enable extensive searching (that is, searching against additional tables, such as an alias table for a person index), you must add the fields from that table to the blocking query.]

[Certain fields used as criteria in the blocking query might contain known default or invalid values. You can define exclusion lists to filter out unwanted or invalid values from the blocking query. For more information, see ["Configuring SBR, Matching, and Blocking Filters"].]]

[[Phonetic Queries in a Master Person Index]

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

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

[[Range Searching]

[Both basic and blocking queries can be configured to perform exact searches or range searches. An exact search performs a query for the exact value entered into a field as search criteria; range searches perform a query on a range of values based on the value entered into a field as search criteria. The basic query supports standard range searching, where both the lower and upper limits of the range is supplied. The blocking query supports standard range searching plus two additional types that use predefined offset values or constants.]

[Offset values allow you to specify values to be added to or subtracted from the entered value to determine the range on which to search. Constants provide a default value to use as a range when no value is entered or when incomplete information is available.]

[Range searching is configured in both midm.xml and query.xml. The processing logic for different types of range searching is described in ["Range Search Processing"].]]

[[The query.xml File]

[The properties for the predefined queries are defined in query.xml. Some of the information entered into the default configuration file is based on the fields you specified for blocking in the wizard, and some is standard across all implementations. For most implementations, this file will require some customization.]

[The following sections provide information about working with query.xml:]

- [[[["Modifying query.xml"]]]
- [[["The query.xml File Structure"]]]]

[[Modifying query.xml]

[You can modify query.xml at any time, but you must regenerate the application and redeploy the project after making any changes to the file. The properties of the blocking query used by the match process should not be modified after moving into production because it can cause unexpected matching weight results. The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes. Most of the components in this file can be configured using the Configuration Editor, which simplifies the process of defining queries by providing a graphical interface to perform the required tasks.]]

[[The query.xml File Structure]

[This section describes the structure of the XML file, including general requirements and constraints, and provides a sample implementation.]

[query.xml File Description]

[["Table 2 query.xml File Structure"] lists each element in query.xml and provides a description of each element along with any requirements or constraints for each element.]

[Table 2 query.xml File Structure]

[Element/Attribute]	[Description]
[QueryBuilderConfig]	[The configuration class for the query builders. This should not be modified.]
[query-builder]	[A list of query definitions. This element defines each query and the attributes of each query.]
[query-builder/name]	[A unique ID for the element. This element is used to identify the Query Builder and is referenced from midm.xml when specifying the query to use on a search page. It is also referenced from mefa.xml when specifying the query to use for matching. No spaces are allowed in this attribute.]
[query-builder/class]	[The fully qualified name of the query class. Two default Query Builder classes are provided.] - [[[com.sun.mdm.index.querybuilder.BasicQueryBuilder - Builds dynamic queries using all the available input fields. When configured to use normalized and phonetic data, this query performs phonetic searches; when configured not to use normalized and phonetic data, this query is used for exact alphanumeric searches.]] - [[com.sun.mdm.index.querybuilder.BlockerQueryBuilder - Builds queries using the criteria defined in the block definitions defined for the query. When a blocking query is performed, the application searches only on the blocks for which the query has complete data.]]]
[query-builder/parser-class]	[The fully qualified name of the class that parses the config elements for each query. This should not be modified for the default queries.]
[query-builder/standardize]	[An indicator of whether the query criteria is standardized before being passed to the query. Specify [**true**] if any fields are standardized for the query; specify [**false**] if no fields are standardized for the query.]

[Element/Attribute]	[Description]
[query-builder/phoneticize]	[An indicator of whether the query criteria is phonetically encoded before being passed to the query. Specify [**true**] if any fields are phonetically encoded for the query specify; [**false**] if no fields are phonetically encoded for the query.]
[config]	[The configuration information for a query. Each query-builder element contains one config element.]
[option]	[One query parameter, specified by key and value attributes, as described below. This is only used by basic queries; blocking queries do not use this element.]
[option/key]	[A parameter for the query option. For the default basic query, only the UseWildCard key is available.]
[option/value]	[The value of the key specified by the corresponding key attribute. For the default option, UseWildCard, specify [**true**] to allow wildcard characters for that query type; otherwise specify [**false**]. When wildcard characters are enabled, you can enter a percent sign (%) to indicate multiple unknown characters.]
[block-definition]	[A list of database hints and defined query criteria blocks, which are identified by unique ID numbers.]
[block-definition/number]	[An attribute of the block-definition element that specifies the unique ID number of each query block. Each block defined for the blocking query must be identified by a unique ID.]
[hint]	[A hint to add to the query to help optimize query execution. Hints are especially useful when a blocking query uses only child object fields; the hint can specify to scan the child object table first. This element is optional. For SQL Server, only OPTION hints are supported.]
[block-rule]	[A list of fields to be included in each query block, including indicators of whether a range is to be used and, if so, what type of range search to perform.]

[Element/Attribute]	[Description]
[type of search]	[An indicator of the type of search to perform on the field defined in the following elements. Each type of search element defines one field in a block-rule element; that is, one field in a query block. This element includes a field element, a source or constant element, and, for range searches only, a default element that defines lower and upper bounds.]
	[Specify one of the following types.]
	■ [[[[equals] - Performs an exact search against either the criteria or the value defined for the constant element.]]
	■ [[[not-equals] - Searches for values that do not equal either the criteria or the value defined for the constant element.]]
	■ [[[greater-than-or-equal] - Performs a search for values that are greater than or equal to either the criteria or the value defined for the constant element.]]
	■ [[[less-than-or-equal] - Performs a search for values that are less than or equal to either the criteria or the value defined for the constant element.]]
	■ [[[range] - Performs a search against a range of either static or user-defined ranges. If you select this option, you must specify upper and lower bounds in a default element.]]]
	[Tip: If a field is to be used for simple range searching (where the user or incoming message supplies lower and upper limits of the range are supplied) be sure to define that field for range searching in midm.xml for the searches that use this query. For more complex range searches that use offset values or constants instead of user-supplied limits, do not define the field for range searching in midm.xml.]
[field]	[The fully qualified field name of the field to be included in the query block (for example, Enterprise.Person.Address.AddressLine1).]
[source]	[The qualified field name of the source field in the object from which the criteria is obtained (for example, Person.Address.AddressLine1). An asterisk (*) can be used as a wildcard character. If the criteria should be a constant value instead of being supplied by a user or incoming message, define a constant element instead of a source element.]
	[[<field>]Enterprise.SystemSBR.Person.Alias.FirstName]
	[[</field>]]
	[[<source>]Person.Alias[*].FirstName</source>]
	[would result in a WHERE clause similar to this:]
	[WHERE Alias.FirstName="Meg" OR Alias.FirstName="Maggie"]
[constant]	[A constant value that provides the criteria for a search. Define this element instead of a source element if the criteria is a constant rather than being user defined. You can use a constant value with the following types of queries: equals, not-equals, greater-than-or-equals, and less-than-or-equals.]
[default]	[A list of upper and lower limits defining a range search. If no limits are defined, the search is a simple range search in which the upper and lower values are supplied by the user or the incoming message (for example, in "Date of Birth From" and "Date of Birth To" fields).]

[Element/Attribute]	[Description]
[lower-bound]	[The lower limit of a constant or offset range search. Use a negative number for the lower limit of an offset search. This number is added to the value supplied for the search to determine the lower limit of the range. The value can be numeric, date, or string. See ["Range Search Processing"] for more information.]
[lower-bound/type]	[The type of range search. Define the type attribute as [**offset**] to use an offset value or as [**constant**] to define a lower constant.]
[upper-bound]	[The upper limit of a constant or offset range search. The value can be numeric, date, or string. See ["Range Search Processing"] for more information.]
[upper-bound/type]	[The type of range search. Define the type attribute as [**offset**] to use an offset value or as [**constant**] to define an upper constant.]

[query.xml Example]

[Below is a sample illustrating the elements in query.xml.]

```
[[<QueryBuilderConfig module-name="QueryBuilder" parser-class=]
[   "com.sun.mdm.index.configurator.impl.querybuilder.QueryBuilderConfiguration">]
[   <query-builder name="ALPHA-SEARCH"]
[    class="com.sun.mdm.index.querybuilder.BasicQueryBuilder"]
[    parser-class="com.sun.mdm.index.configurator.impl.querybuilder.]
[    KeyValueConfiguration" standardize="true" phoneticize="false">]
[      <config>]
[         <option key="UseWildcard" value="true"/>]
[      </config>]
[   </query-builder>]
[   <query-builder name="PHONETIC-SEARCH"]
[    class="com.sun.mdm.index.querybuilder.BasicQueryBuilder"]
[    parser-class="com.sun.mdm.index.configurator.impl.querybuilder.]
[    KeyValueConfiguration" standardize="true" phoneticize="true">]
[      <config>]
[         <option key="UseWildcard" value="false"/>]
[      </config>]
[   </query-builder>]
[   <query-builder name="BLOCKER-SEARCH"]
[    class="com.sun.mdm.index.querybuilder.BlockerQueryBuilder" parser-]
[    class="com.sun.mdm.index.configurator.impl.blocker.BlockerConfig"]
[    standardize="true" phoneticize="true">]
[      <config>]
[         <block-definition number="ID000000">]
[            <block-rule>]
[               <equals>]
[                   <field>Enterprise.SystemSBR.Person.FnamePhonetic]
[                   </field>]
[                   <source>Person.FnamePhoneticCode</source>]
[               </equals>]
[               <equals>]
[                   <field>Enterprise.SystemSBR.Person.LnamePhonetic]
[                   </field>]
[                   <source>Person.LnamePhoneticCode</source>]
[               </equals>]
[            </block-rule>]
[         </block-definition>]
[         <block-definition number="ID000001">]
[            <block-rule>]
[               <equals>]
```

```
[                    <field>Enterprise.SystemSBR.Person.SSN</field>]
[                        <source>Person.SSN</source>]
[                </equals>]
[            </block-rule>]
[        </block-definition>]
[        <block-definition number="ID000002">]
[            <hint>ALL_ROWS</hint>]
[            <block-rule>]
[                <equals>]
[                    <field>Enterprise.SystemSBR.Person.FnamePhonetic]
[                    </field>]
[                    <source>Person.FnamePhoneticCode</source>]
[                </equals>]
[                <range>]
[                    <field>Enterprise.SystemSBR.Person.DOB</field>]
[                    <source>Person.DOB</source>]
[                    <default>]
[                        <lower-bound type="offset">-5</lower-bound>]
[                        <upper-bound type="offset">5</upper-bound>]
[                    </default>]
[                </range>]
[                <equals>]
[                    <field>Enterprise.SystemSBR.Person.Gender</field>]
[                    <source>Person.Gender</source>]
[                </equals>]
[            </block-rule>]
[        </block-definition>]
[    </config>]
[  </query-builder>]
[</QueryBuilderConfig>]
[]]]]
```

[[Range Search Processing]

[Both basic and blocking queries can be configured to perform both exact searches and range searches. The following topics describe how different configurations of exact and range searches are processed.]

- [[[["Basic Query Range Searching"]]]]

- [[[["Blocking Query Range Searching"]]]]]

[[Basic Query Range Searching]

[Range searching for basic queries is configured in the search page section of midm.xml by tagging the field with a "choice" attribute. When you specify a field for range searching, two corresponding fields appear on the MIDM with "From" and "To" appended to the name (for example, a field named "Date of Birth" would display two fields: "Date of Birth From" and "Date of Birth To"). You can also define a field for both exact and range searching by defining the field twice for the search page, once with the choice attribute set to "exact" and once with it set to "range". In this case, three fields appear on the MIDM: one with the given field name, one with "From" appended to the name, and one with "To" appended to the name.]

[[["Table 3 Standard Range Queries"] describes the queries formed for different exact or range search scenarios. ["Table 4 Combination Exact and Range Queries"] describes the queries formed for combination exact and range search scenarios.]

[The following variables are used in these tables:]

- [[[field_name is the field name as specified in the search page section of midm.xml (the field named field_name is used for exact searching)]]
- [[value is the value entered into the exact search field]]
- [[value_from is the value entered into the field_name From field]]
- [[value_to is the value entered into the field_name To field]]]

[Table 3 Standard Range Queries]

[Field Configuration in midm.xml]	[Resulting Fields on MIDM]	[Fields Populated for Search]	[Where Clause]
[[**choice**] attribute set to "exact"]	[field_name]	[field_name]	[where [**field_name**] = [**value**]]
[[**choice**] attribute set to "'range"]	[[**field_name**] From] [[**field_name**] To]	[[**field_name**] From] [[**field_name**] To]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**value_to**]]
[[**choice**] attribute set to "range"]	[[**field_name**] From] [[**field_name**] To]	[[**field_name**] From]	[where [**field_name**] >= [**value_from**]]
[[**choice**] attribute set to "range"]	[[**field_name**] From] [[**field_name**] To]	[[**field_name**] To]	[where [**field_name**] <= [**value_to**]]

[In the following table, when field_name is populated but not used in the WHERE clause, its value is used for weighting purposes. These cases are marked with an asterisk (*).]

[Table 4 Combination Exact and Range Queries]

[Field Configuration in midm.xml]	[Resulting Fields on MIDM]	[Fields Populated for Search]	[Where Clause]
[field defined once with [**choice**] attribute set to "exact" and once with it set to "range"]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name]	[where [**field_name**] = [**value**]]
[field defined once with [**choice**] attribute set to "exact" and once with it set to "range"]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] From] [[**field_name**] To]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**value_to**]]
[field defined once with [**choice**] attribute set to "exact" and once with it set to "range"]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] From]	[where [**field_name**] >= [**value_from**]]
[field defined once with [**choice**] attribute set to "exact" and once with it set to "range"]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] To]	[where [**field_name**] <= [**value_to**]]
[field defined once with [**choice**] attribute set to "exact" and once with it set to "range" *]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] From]	[where [**field_name**] >= [**value_from**]]
[field defined once with [**choice**] attribute set to "exact" and once with it set to "range" *]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] To]	[where [**field_name**] <= [**value_to**]]

[Field Configuration in midm.xml]	[Resulting Fields on MIDM]	[Fields Populated for Search]	[Where Clause]
[field defined once with [**choice**] attribute set to "exact" and once with it set to "range" *]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] From] [[**field_name**] To]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**value_to**]]

[[Blocking Query Range Searching]

[Blocking queries are configured in query.xml, and, if the blocking query is used on the MIDM, in midm.xml. In order for the fields defined for range searching in the blocking query to appear on the MIDM, the fields must be configured correctly in midm.xml.]

[In addition to the standard range searching (described under [Basic Query Range Searching]), blocking queries support constant and offset range searches, allowing you to specify default upper and lower offset values or to specify upper and lower constant limits. Using offsets adds the specified values to the actual field value to determine the range on which to search. Note that this means the lower offset value should be a negative number and the upper offset value should be a positive number in order to create a valid range. You can also define a combination of a constant upper limit with lower offset value or a constant lower limit with an upper offset value.]

[Blocking Query Offset Values]

[When upper and lower offset values are defined, the application searches for values that are greater than or equal to the field value plus the lower offset value (which is typically a negative number) and less than or equal to the field value plus the upper offset value. You do not need to define both an upper and a lower offset value.]

> **[Note]:** [For date fields, the method for adding the offsets is different for numeric than for date type fields. For numeric data types, the offset value is added to the actual number. For date data types, the offset value is added to the day portion of the date (for example, if the offsets were -5 and +5 and the date entered is 01/10/2010, then the upper and lower bounds would be 01/05/2010 and 01/15/2010).]

[["Table 5 Standard Offset Range Queries"] describes the queries formed for different exact or range offset search scenarios. ["Table 6 Combination Offset Range Queries"] describes the query formed for combination exact and offset range search scenarios.]

[The following variables are used in these tables:]

- [[[field_name is the field name as specified in the search page section of midm.xml (the field named field_name is used for exact searching)]]

- [[value is the value entered into the exact search field]]

- [[value_from is the value entered into the field_name From field]]

- [[value_to is the value entered into the field_name To field]]

- [[lower is the lower offset value]]

- [[upper is the upper offset value]]]

[Table 5 Standard Offset Range Queries]

[Field Configuration in midm.xml]	[Resulting Fields on MIDM]	[Offset Configuration in query.xml]	[Fields Populated for Search]	[Where Clause]
[[**choice**] attribute set to "exact"]	[field_name]	[both upper and lower offsets defined]	[field_name]	[where [**field_name**] >= ([**value**] + [**lower**]) and [**field_name**] <= ([**value**] + [**upper**])]
[[**choice**] attribute set to "exact"]	[field_name]	[only lower offset defined]	[field_name]	[where [**field_name**] >= ([**value**] + [**lower**])]
[[**choice**] attribute set to "exact"]	[field_name]	[only upper offset defined]	[field_name]	[where [**field_name**] <= ([**value**] + [**upper**])]
[[**choice**] attribute set to "range"]	[[**field_name**] From] [[**field_name**] To]	[upper, lower, or both offsets are defined]	[[**field_name**] From] [[**field_name**] To]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**value_to**]]
[[**choice**] attribute set to "range"]	[[**field_name**] From] [[**field_name**] To]	[upper, lower, or both offsets are defined]	[[**field_name**] From]	[where [**field_name**] >= [**value_from**]]
[[**choice**] attribute set to "range"]	[[**field_name**] From] [[**field_name**] To]	[upper, lower, or both offsets are defined]	[[**field_name**] To]	[where [**field_name**] <= [**value_to**]]

[In ["Table 6 Combination Offset Range Queries"], the field configuration in midm.xml defines the field twice for searching, once with the choice attribute set to "exact" and once with it set to "range".]

[In the following cases, when field_name is populated but not used in the WHERE clause, its value is used for weighting purposes. These cases are marked with an asterisk (*).]

[Table 6 Combination Offset Range Queries]

[Offset Configuration in query.xml]	[Fields on MIDM]	[Fields Populated for Search]	[Query Result]
[both upper and lower bound offsets are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name]	[where [**field_name**] >= ([**value**] + [**lower**]) and [**field_name**] <= ([**value**] + [**upper**])]
[only a lower offset is defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name]	[where [**field_name**] >= ([**value**] + [**lower**])]
[only an upper offset is defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name]	[where [**field_name**] <= ([**value**] + [**upper**])]
[upper, lower, or both offsets are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] From] [[**field_name**] To]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**value_to**]]

[Offset Configuration in query.xml]	[Fields on MIDM]	[Fields Populated for Search]	[Query Result]
[upper, lower, or both offsets are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] From]	[where [**field_name**] >= [**value_from**]]
[upper, lower, or both offsets are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] To]	[where [**field_name**] <= [**value_to**]]
[both upper and lower offsets are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] From]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= ([**value**] + upper)]
[only a lower offset is defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] From]	[where [**field_name**] >= ([**value**] + [**lower**])]
[only an upper offset is defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] From]	[where [**field_name**] <= ([**value**] + [**upper**])]
[both upper and lower offsets are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] To]	[where [**field_name**] <= [**value_to**] and [**field_name**] >= ([**value**] + [**lower**])]
[only a lower offset is defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] To]	[where [**field_name**] >= ([**value**] + [**lower**])]
[only an upper offset is defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] To]	[where [**field_name**] <= ([**value**] + [**upper**])]
[both upper and lower offsets are defined*]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] From] [[**field_name**] To]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**value_to**]]

[Blocking Query Constants]

[When you define upper and lower constants for a field, these values are used for the WHERE clause of the query if no data is passed in as search criteria for that field. They are also used when only one of the "from" or "to" fields is populated. You do not need to define both an upper and a lower constant value. If you define only an upper constant value, only a "less than or equals" clause is used in the query; if you define only a lower constant value, only a "greater than or equals" clause is used in the query.]

> **[Note]:** [For numeric type fields, the constant must be defined as all digits, with one decimal point allowed. For date type fields, the constant must be in the standard SQL format of yyyy-mm-dd.]

[["Table 7 Standard Constant Range Queries"] describes the queries formed for different exact or range constant search scenarios. ["Table 8 Combination Constant

Range Queries"] describes the query formed for combination exact and range search scenarios.]

[The following variables are used in these tables:]

- [[[field_name is the field name as defined in the search page section of midm.xml (the field named field_name is used for exact searching)]]]

- [[value is the value entered into the exact search field]]

- [[value_from is the value entered into the field_name From field]]

- [[value_to is the value entered into the field_name To field]]

- [[lower is the lower constant value]]

- [[upper is the upper constant value]]]

[Table 7 Standard Constant Range Queries]

[Field Configuration in midm.xml]	[Resulting Fields on MIDM]	[Fields Populated for Search]	[Where Clause]
[[**choice**] attribute set to "exact"]	[field_name]	[field_name]	[where [**field_name**] = [**value**]]
[[**choice**] attribute set to "range"]	[[**field_name**] From] [[**field_name**] To]	[[**field_name**] From] [[**field_name**] To]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**value_to**]]
[[**choice**] attribute set to "range"]	[[**field_name**] From] [[**field_name**] To]	[[**field_name**] From]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**upper**]]
[[**choice**] attribute set to "range"]	[[**field_name**] From] [[**field_name**] To]	[[**field_name**] To]	[where [**field_name**] <= [**value_to**] and [**field_name**] >= [**lower**]]

[In ["Table 8 Combination Constant Range Queries"], the field configuration in midm.xml defines the field twice for searching, once with the choice attribute set to "exact" and once with it set to "range".]

[In the following cases, when field_name is populated but not used in the WHERE clause, its value is used for weighting purposes. These cases are marked with an asterisk (*).]

[Table 8 Combination Constant Range Queries]

[Offset Configuration in query.xml]	[Fields on MIDM]	[Fields Populated for Search]	[Query Result]
[upper, lower, or both constants are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name]	[where [**field_name**] = [**value**]]
[upper, lower, or both constants are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] From] [[**field_name**] To]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**value_to**]]

[Offset Configuration in query.xml]	[Fields on MIDM]	[Fields Populated for Search]	[Query Result]
[either upper or both constants are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] From]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**upper**]]
[lower constant is defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] From]	[where [**field_name**] >= [**value_from**]]
[either upper or both constants are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] From]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**upper**]]
[lower constant is defined *]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] From]	[where [**field_name**] >= [**value_from**]]
[either lower or both constants are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] To]	[where [**field_name**] <= [**value_to**] and [**field_name**] >= [**lower**]]
[upper constant is defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] To]	[where [**field_name**] <= [**value_to**]]
[either lower or both constants are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] To]	[where [**field_name**] <= [**value_to**] and [**field_name**] >= [**lower**]]
[upper constant is defined *]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] To]	[where [**field_name**] <= [**value_to**]]
[upper, lower, or both constants are defined *]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] From] [[**field_name**] To]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**value_to**]]

[Blocking Query Offset and Constant Combinations]

[You can use a combination of offset and constant values to define range searching for a field. ["Table 9 Combination Constant and Offset Range Queries"] describes the query formed for combination offset and constant search scenarios.]

[The following variables are used in these tables:]

- [[[field_name is the field name as defined in the search page section of midm.xml (the field named field_name is used for exact searching)]]]
- [[value is the value entered into the exact search field]]
- [[value_from is the value entered into the field_name From field]]
- [[value_to is the value entered into the field_name To field]]
- [[lower is the lower constant or offset value]]
- [[upper is the upper constant or offset value]]]

[In ["Table 9 Combination Constant and Offset Range Queries"], the field configuration in midm.xml defines the field twice for searching, once with the choice attribute set to "exact" and once with it set to "range".]

[In the following cases, when field_name is populated but not used in the WHERE clause, its value is used for weighting purposes. These cases are marked with an asterisk (*).]

[Table 9 Combination Constant and Offset Range Queries]

[Offset Configuration in query.xml]	[Fields on MIDM]	[Fields Populated for Search]	[Query Result]
[upper offset and lower constant are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name]	[where [**field_name**] >= [**lower**] and [**field_name**] <= ([**value**] + [**upper**])]
[upper offset and lower constant are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] From] [[**field_name**] To]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**value_to**]]
[upper offset and lower constant are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] From]	[where [**field_name**] >= [**value_from**]]
[upper offset and lower constant are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] To]	[where [**field_name**] <= [**value_to**] and [**field_name**] >= [**lower**]]
[upper offset and lower constant are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] From]	[where [**field_name**] <= ([**value**] + [**upper**]) and [**field_name**] >= [**value_from**]]
[upper offset and lower constant are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] To]	[where [**field_name**] <= [**value_to**] and [**field_name**] >= [**lower**]]
[upper offset and lower constant are defined *]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] From] [[**field_name**] To]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**value_to**]]
[upper constant and lower offset are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name]	[where [**field_name**] <= [**upper**] and [**field_name**] >= ([**value + lower**])]
[upper constant and lower offset are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] From] [[**field_name**] To]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**value_to**]]
[upper constant and lower offset are defined]	[field_na] [me] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] From]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**upper**]]

[Offset Configuration in query.xml]	[Fields on MIDM]	[Fields Populated for Search]	[Query Result]
[upper constant and lower offset are defined]	[field_name] [[**field_name**] From] [[**field_name**] To]	[[**field_name**] To]	[where [**field_name**] <= [**value_to**]]
[upper constant and lower offset are defined *]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] From]	[where [**field_name**] <= [**upper**] and [**field_name**] >= [**value_from**]]
[upper constant and lower offset are defined *]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] To]	[where [**field_name**] <= [**value_to**] and [**field_name**] >= ([**value**] + [**lower**])]
[upper constant and lower offset are defined +]	[field_name] [[**field_name**] From] [[**field_name**] To]	[field_name] [[**field_name**] From] [[**field_name**] To]	[where [**field_name**] >= [**value_from**] and [**field_name**] <= [**value_to**]]

[[Configuring Manager Service Components]

[In master.xml, you define certain system parameters for the Manager Service, such as matching thresholds, EUID properties, and the blocking query to use for match processing. The Manager Service is the main interface of the indexing system. This interface coordinates all components of the master person index application, including the database, master person index project, Master Index Data Manager, runtime environment, and match engine. The main interface is a stateless session bean, though some methods return objects that have handles to stateful beans.]

[The following sections describe the Manager Service and master.xml.]

- [[[["Manager Service Components"]]]]
- [[["The master.xml File"]]]]

[[Manager Service Components]

[In master.xml, you define certain properties of the match process, such as duplicate and match thresholds, the query to use for matching, logic for automatic merges, and properties of the EUIDs assigned by the master person index application (such as their length and whether a checksum value is used). This file is also used to define the update mode (optimistic or pessimistic) and merged record updates.]

[The following Manager Service components are configured by master.xml:]

- [[[["Master Controller Configuration"]]]]
- [[["Decision Maker"]]]]
- [[["EUID Generator"]]]]

[[Master Controller Configuration]

[The MasterControllerConfig element of master.xml controls four components of the matching and update process.]

- [[[["Custom Logic Classes in master.xml"]]]]
- [[["Update Mode in master.xml"]]]]

- [[["Merged Record Updates in master.xml"]]]
- [[["Blocking Query in master.xml"]]]
- [[["Transactional Support"]]]]

[Custom Logic Classes in master.xml]

[Custom logic classes specify any custom plug-ins created for the master person index project that define custom processing for the execute match methods. If no classes are specified, execute match processing is carried out using the default logic (this is described in [*Understanding Sun Master Index Processing*]).]

[Update Mode in master.xml]

[The update mode specifies whether a record's potential duplicate list is reevaluated when key fields are updated in the record. Performing the reevaluation helps keep the potential duplicate list current, but requires more system resources.]

[There are two update modes.]

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

[Merged Record Updates in master.xml]

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

[Blocking Query in master.xml]

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

[Transactional Support]

[Oracle Healthcare Master Person Index supports local and distributed transaction processing. You can configure the master person index application to distribute transactions across applications, to distribute transactions only within the master person index application, or to not use distributed transactions at all. This is defined in the transaction element.]]

[[Decision Maker]

[The DecisionMakerConfig element of master.xml allows you to specify how the Manager Service evaluates query results. For the default Decision Maker, you can configure these parameters:]

- [[[["OneExactMatch"]]]
- [[[["SameSystemMatch"]]]
- [[[["DuplicateThreshold"]]]
- [[[["MatchThreshold"]]]]

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

[OneExactMatch]

[This parameter specifies logic for assumed matches. If OneExactMatch is set to true and there is more than one record above the match threshold, then none of the records are considered an assumed match and all are flagged as potential duplicates. If OneExactMatch is set to false and there is more than one record above the match threshold, then the record with the highest matching weight is considered an assumed match and the rest are flagged as potential duplicates.]

[SameSystemMatch]

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

[DuplicateThreshold]

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

[MatchThreshold]

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

[[EUID Generator]

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

- [[[["IdLength"]]]
- [[[["ChecksumLength"]]]
- [[[["ChunkSize"]]]]

[IdLength]

[This parameter defines the length of the EUIDs created by the master person index application. By default, the length of the EUID columns in the master person index database is 20. If you choose an ID length larger than 20, make sure to manually modify the length of the EUID columns in the database creation scripts.]

[ChecksumLength]

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

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

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

[ChunkSize]

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

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

[[The master.xml File]

[The properties of the Manager Service are defined in master.xml. The information entered into the default configuration file is standard across all implementations, so the file will require some customization.]

[The following sections provide information about working with master.xml:]

- [[[["Modifying master.xml"]]]]
- [[["The master.xml File Structure"]]]]]

[[Modifying master.xml]

[You can modify master.xml at any time, but you must regenerate the application and redeploy the project after making any changes to the file. Use caution when updating this file after moving into production, since changing certain properties, such as the blocking query, can cause unexpected matching and weighting results. Most of the configuration options in this file cannot be modified using the Configuration Editor. The exceptions are the match and duplicate thresholds. The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes.]]

[[The master.xml File Structure]

[This topic describes the structure of the XML file, general requirements, and constraints. It also provides a sample implementation.]

[master.xml File Description]

[["Table 10 master.xml File Structure"] lists each element in master.xml and provides a description of each element along with any requirements or constraints for each element.]

[Table 10 master.xml File Structure]

[Element/Attribute]	[Description]
[MasterControllerConfig]	[The configuration class for the Manager Service. The attributes define the module name and Java class. The default values should not be changed.]
[logic-class]	[A custom plug-in that defines custom processing logic for the execute match functions that can be called from client applications. This element is optional.]
[logic-class-gui]	[A custom plug-in that defines custom processing logic for the execute match function that is called from the Master Index Data Manager (MIDM). This element is optional.]
[update-mode]	[An indicator of whether to recalculate potential duplicates when a record is updated. Specify Pessimistic to recalculate potential duplicates; specify [**Optimistic**] to prevent potential duplicate recalculation on updates.]
[merged-record-update]	[An indicator of whether records with a status of Merged can be updated. Specify [**Enabled**] to allow updates of merged records; specify [**Disabled**] to ensure that records with a Merged status are not updated.]
[execute-match]	[Specifies the blocking query to use for match processing.]
[query-builder]	[The name of the blocking query to use for match processing. The name must match a query defined in query.xml.]
[option]	[Optional parameters for the blocking query. Currently parameters are not used by any predefined blocking queries.]
[option/key]	[A parameter for the blocking query.]
[option/value]	[The value of the key specified by the corresponding key attribute.]

[Element/Attribute]	[Description]
[transaction]	[The transaction mode for the master person index application. Specify one of the following values:]
	▪ [[[LOCAL - Transactions are not distributed.]]
	▪ [[CONTAINER - Transactions are distributed across applications.]]
	▪ [[BEAN - Transactions are distributed within the master person index application.]]]
[DecisionMakerConfig]	[The configuration class for the Decision Maker. The attributes define the module name and Java class. The default values should not be changed.]
[decision-maker-class]	[The Java class that contains the methods used by the Decision Maker class. The default value, com.sun.mdm.index.decision.impl.DefaultDecisionMaker, should not need to be changed, but you can implement a custom Decision Maker class. The default class accepts the parameters described below.]
[parameters]	[A list of parameters for the Decision Maker class.]
[parameter]	[A definition of a Decision Maker parameter. The parameters element can contain multiple parameter elements, each defining one parameter.]
[description]	[A brief description of the parameter. This element is optional.]
[parameter-name]	[The name of the parameter. The default Decision Maker class takes the following parameters (see ["Decision Maker" on page 2-27] for more information about these parameters).]
	▪ [[[[OneExactMatch] - A Boolean indicator of whether an assumed match is made when there are more than one record above the match threshold.]]
	▪ [[[SameSystemMatch] - A Boolean indicator of whether an assumed match can be made between two records that originate from the same external system.]]
	▪ [[[DuplicateThreshold] - The lowest match weight at which two records are considered to be potential duplicates.]]
	▪ [[[MatchThreshold] - The lowest match weight at which two records are assumed to be a match of one another.]]]
[parameter-type]	[The type of parameter. Valid values are java.lang.Long, java.lang.Short, java.lang.Byte, java.lang.String, java.lang.Integer, java.lang.Boolean, java.lang.Double, or java.lang.Float.]
[parameter-value]	[The value of the parameter. For OneExactMatch and SameSystemMatch, this must be a Boolean value. For MatchThreshold and DuplicateThreshold, this must be a Float value.]
[EuidGeneratorConfig]	[The configuration class for the EUID Generator. The attributes define the module name and Java class. The default values should not be changed.]
[euid-generator-class]	[The Java class used by the master person index application to generate new EUIDs. The default class is com.sun.mdm.index.idgen.impl.DefaultEuidGenerator, which assigns sequential EUIDs based on the three parameters described below.]
[parameters]	[A list of parameters for the EUID Generator class.]
[parameter]	[A parameter definition. The parameters element can contain multiple parameter elements, each defining one parameter.]

[Element/Attribute]	[Description]
[description]	[A brief description of the parameter. This element is optional.]
[parameter-name]	[The name of the parameter. The default EUID Generator class takes the following parameters (see ["EUID Generator"] for more information about these parameters).] ■ [[[[**IdLength**] - The length of the EUIDs generated by the master person index application.]] ■ [[[**CheckSum**] - The length of the checksum value used to validate EUIDs.]] ■ [[[**ChunkSize**] - The number of EUIDs allocated to the server at one time.]]]
[parameter-type]	[The type of parameter. Valid values are java.lang.Long, java.lang.Short, java.lang.Byte, java.lang.String, java.lang.Integer, java.lang.Boolean, java.lang.Double, or java.lang.Float.]
[parameter-value]	[The value of the parameter. For the default parameters, the values are all integers.]

[master.xml Example]

[Below is a sample of master.xml configuration.]

```
[[<MasterControllerConfig module-name="MasterController" parser-class=]
[ "com.sun.mdm.index.configurator.impl.master.MasterControllerConfiguration">]
[    <logic-class>CustomMatchLogic</logic-class>]
[    <logic-class-gui>CustomMatchLogicMIDM</logic-class-gui>]
[    <update-mode>Pessimistic</update-mode>]
[    <merged-record-update>Disabled</merged-record-update>]
[    <execute-match>]
[        <query-builder name="BLOCKER-SEARCH"></query-builder>]
[    </execute-match>]
[</MasterControllerConfig>]
[<DecisionMakerConfig module-name="DecisionMaker" parser-class=]
[ "com.sun.mdm.index.configurator.impl.decision.DecisionMakerConfiguration">]
[    <decision-maker-class>]
[        com.sun.mdm.index.decision.impl.DefaultDecisionMaker]
[    </decision-maker-class>]
[    <parameters>]
[        <parameter>]
[            <parameter-name>OneExactMatch</parameter-name>]
[            <parameter-type>java.lang.Boolean</parameter-type>]
[            <parameter-value>false</parameter-value>]
[        </parameter>]
[        <parameter>]
[            <parameter-name>SameSystemMatch</parameter-name>]
[            <parameter-type>java.lang.Boolean</parameter-type>]
[            <parameter-value>true</parameter-value>]
[        </parameter>]
[        <parameter>]
[            <parameter-name>DuplicateThreshold</parameter-name>]
[            <parameter-type>java.lang.Float</parameter-type>]
[            <parameter-value>7.25</parameter-value>]
[        </parameter>]
[        <parameter>]
[            <parameter-name>MatchThreshold</parameter-name>]
[            <parameter-type>java.lang.Float</parameter-type>]
[            <parameter-value>29.0</parameter-value>]
[        </parameter>             ]
[    </parameters>]
```

```
[</DecisionMakerConfig>]
[<EuidGeneratorConfig module-name="EuidGenerator" parser-class=]
["com.sun.mdm.index.configurator.impl.idgen.EuidGeneratorConfiguration">]
[    <euid-generator-class>]
[        com.sun.mdm.index.idgen.impl.DefaultEuidGenerator]
[    </euid-generator-class>]
[    <parameters>]
[        <parameter>]
[            <parameter-name>IdLength</parameter-name>]
[            <parameter-type>java.lang.Integer</parameter-type>]
[            <parameter-value>10</parameter-value>]
[        </parameter>]
[        <parameter>]
[            <parameter-name>ChecksumLength</parameter-name>]
[            <parameter-type>java.lang.Integer</parameter-type>]
[            <parameter-value>0</parameter-value>]
[        </parameter>]
[        <parameter>]
[            <parameter-name>ChunkSize</parameter-name>]
[            <parameter-type>java.lang.Integer</parameter-type>]
[            <parameter-value>1000</parameter-value>]
[        </parameter>]
[    </parameters>]
[</EuidGeneratorConfig>]
[]]]]]
```

[[Configuring Match Fields]

[The Matching Service, configured in mefa.xml, contains the matching and standardization engines used in the match process, as well as the phonetic encoders used for phonetically encoding data. You can configure the match and standardization engines for the master person index application in mefa.xml, and also specify special standardization, matching, and weighting logic used by the engines. This file also defines the strategy for identifying unique records and finding the best matches in the master person index database. For optimization, the Match Field components are configurable, allowing you to choose the strategy that best fits your requirements or to implement your own custom components.]

[The following sections describe the components of the Matching Service and the structure of mefa.xml:]

- [[[["Matching Service Components"]]]]

- [[["Sample Standardization and Matching Sequence"]]]]

- [[["The mefa.xml File"]]]]

[[Matching Service Components]

[The Matching Service is configured by mefa.xml, which defines the configurable properties for standardizing data and matching records. These processes are highly configurable for the master person index application, allowing you to design and develop the match strategy that best suits your processing requirements.]

[The following components make up the Matching Service:]

- [[[["Standardization Configuration"]]]]

- [[["Matching Configuration"]]]]

- [[["MEFA Configuration"]]]]

- [[["Phonetic Encoders"]]]]

[[Standardization Configuration]

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

[Data Reformatting]

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

[Data Normalization]

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

[Phonetic Encoding]

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

[[Matching Configuration]

[The MatchingConfig section of mefa.xml allows you to define the data fields that are sent to the match engine (called the [**match string**]). Probabilistic weighting is performed only against the fields you specify as the match columns. You can specify any field in the object structure as a match column as long as the is configured to use all fields specified. You must specify at least one match field. You can further configure the match string by removing known default or invalid values from the matching process. For more information, see ["Configuring SBR, Matching, and Blocking Filters"].]

[The configuration of this section of mefa.xml is specific to the you are using and the types of fields on which you are matching. For more information about how the matching should be configured for the OHMPI Match Engine, see [*Oracle Healthcare Master Person Index Match Engine Reference*] (Part Number E18470-01).]]

[[MEFA Configuration]

[The MEFAConfig section specifies the Java classes to be used by components of the Matching Service, including the match and standardization engines, block picker, and pass controller. The match and standardization engines control the processes of standardizing data and generating matching probability weights between records. The block picker and pass controller define how the blocking query is executed during the match process.]

[Match and Standardization Engines]

[Oracle Healthcare Master Person Index provides the ability to use the standardization and match engines that best suit your indexing requirements. You can configure the master person index application to use the OHMPI Match Engine and the OHMPI Standardization Engine, or you can configure the index to use a customized engine of your choice.]

[These engines perform two functions:]

- [[[Standardize data to a common format]]

- [[Calculate the likelihood that two objects match]]]

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

[Block Picker and Pass Controller]

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

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

[[Phonetic Encoders]

[Oracle Healthcare Master Person Index provides extensible phonetic encoding capabilities, which are typically used to retrieve records with similar field values from the database for matching. By default, several phonetic encoders are defined to be used in the master person index application. Typically, Soundex is used to encode first names (or SoundexFR for first names in the France national domain) and NYSIIS to encode last names. When using the OHMPI Standardization Engine, you can specify different types of phonetic encoders, such as Metaphone, Double Metaphone, and Refined Soundex. When you specify the fields in the standardization configuration to be phonetically encoded, you can select one of the encoders defined in the phonetic encoders section.]]]

[[Sample Standardization and Matching Sequence]

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

1. [[[A record is received from an external system.]]

2. [[The local ID does not yet exist in the master person index application; initiate the standardization and matching process.]]

3. [[Standardize the record to a common format.]]

4. [[Standardize free-form text.]]

5. [[Normalize fields that need to be converted to a common format.]]

6. [[Phonetically encode fields that are commonly misspelled or spelled in different ways.]]

7. [[Match the record against entries in the database.]]

8. [[Use the selected blocking query (specified in master.xml) to retrieve a block of records that might match the new record.]]

9. [[Build and execute the query according to the input record.]]

10. [[Calculate match scores comparing the incoming record against existing records (this is done by the match engine).]]

11. [[Determine whether to repeat the matching process with another block of records, based on the MEFAConfig element in mefa.xml.]]

12. [[Return match scores for further processing.]]

13. [[Determine whether to add the system record to an existing EUID record or to insert the system record as a new EUID record (based on the parameters defined in the DecisionMaker element of master.xml).]]]]

[[The mefa.xml File]

[The properties for the match and standardization process are defined in mefa.xml. Some of the information entered into the default configuration file is taken from the wizard, but the file might require additional customization in order to meet your data processing needs.]

[The following sections provide information about working with the mefa.xml:]

- [[[["Modifying master.xml"]]]

- [[["The master.xml File Structure"]]]]

[[Modifying mefa.xml]

[You can modify mefa.xml at any time, but modifying the file is not recommended once you move to production because this file defines how records are processed and data integrity is maintained. You must regenerate the application and redeploy the project after making any changes to this file. Modifying this file once you are in production might cause weighting and standardization to be handled differently, causing unexpected match weight results.]

[Most of the components configured by this file can be modified using the Configuration Editor. The editor provides a graphical interface that simplifies defining normalization, standardization, matching, and phonetic encoding. It also maintains referential integrity between files in cases where standardization, normalization, or phonetic encoding requires additional fields to be added to the object structure. The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes.]]

[[The mefa.xml File Structure]

[This section describes the structure of the XML file, general requirements, and constraints. It also provides a sample implementation.]

[mefa.xml Description]

[["Table 11 mefa.xml File Structure"] lists each element in mefa.xml and provides a description of each element along with any requirements or constraints for each element.]

[Table 11 mefa.xml File Structure]

[Element/Attribute]	[Description]
[StandardizationConfig]	[The configuration information for fields to be standardized. It consists of several structures that define standardization rules for a set of fields. The StandardizationConfig attributes define the module name and Java class, and their default values should not be changed.]
[standardize-system-object]	[A standardization structure that defines configuration rules, including normalization, parsing, and phonetic encoding. Each standardization structure contains three primary elements: structures-to-normalize, free-form-texts-to-standardize, and phoneticize-fields. These elements are all required, however any of them can be empty.]
[system-object-name]	[The name of the object containing the fields defined for standardization. Specifying the parent object allows you to specify any field in any object for standardization. You can also create multiple standardization structures and specify a different object for each structure.]
[structures-to-normalize]	[The configuration information for fields that require normalization (but not parsing or reformatting) before being processed by the standardization engine.]
[group/standardization-type]	[The type of standardization to perform on the source fields. This is specific to the type of data being processed and the standardization engine being used. For more information about OHMPI Standardization Engine types, see [*Oracle Healthcare Master Person Index Standardization Engine Reference*] (Part Number E18471-01).]
[group/domain-selector]	[The Java class used by the OHMPI Standardization Engine to determine the nationality of the data being processed. If no selector is specified, the default is US.]
	[Possible values for the OHMPI Standardization Engine include the following:]
	■ [[[com.sun.mdm.index.matching.impl.SingleDomainSelectorAU]]
	■ [[com.sun.mdm.index.matching.impl.SingleDomainSelectorFR]]
	■ [[com.sun.mdm.index.matching.impl.SingleDomainSelectorMX]]
	■ [[com.sun.mdm.index.matching.impl.SingleDomainSelectorUK]]
	■ [[com.sun.mdm.index.matching.impl.SingleDomainSelectorUS]]
	■ [[com.sun.mdm.index.matching.impl.MultipleDomainSelector]]]

[Element/Attribute]	[Description]
[local-field-name]	[The ePath to an identifying field in the object structure that indicates which of the defined local-codes definitions to use. If no field is specified, the standardization engine defaults to the United States domain. This field must be contained in the object that contains the fields defined for normalization in this structure.]
[locale-maps]	[A list of local codes that define how the standardization engine determines which national domain to use.]
[local-codes]	[A list of value and locale pairs that indicate the national domain to use based on the value of the identifying field in an incoming message (specified by the local-field-name).]
[value]	[A value that, when contained in the identifying field, indicates that the standardization engine will use the corresponding locale element to determine which national domain to use to standardize the data. To specify a default domain, enter "Default" in this element.]
[locale]	[A domain code indicating which national domain to use to standardize data when the identifying field value in a transaction matches the corresponding value element.]
	[The supported locale codes for the OHMPI Standardization Engine include the following:]
	■ [[[[**AU**] - for Australian data]]
	■ [[[**FR**] - for French data]]
	■ [[[**MX**] - for Mexican data]]
	■ [[[**UK**] - for United Kingdom data]]
	■ [[[**US**] - for United States data]]]
[unnormalized-source-fields]	[A list of source fields to be normalized.]
[source-mapping]	[The configuration information for one field in the list of source fields to be normalized.]
[unnormalized-source-field-name]	[The ePath of the source field to normalize in the system object (for example, Person.FirstName).]
[standardized-object-field-id]	[An identification code that identifies the field to normalize to the standardization engine. This ID is specific to the standardization engine in use and must correspond to a field ID defined by that engine. For more information, see [*Oracle Healthcare Master Person Index Standardization Engine Reference*] (Part Number E18471-01).]
[normalization-targets]	[A list of destination fields to hold the normalized data.]
[target-mapping]	[The configuration information for one field in the list of destination fields.]
[freeform-texts-to-standardize]	[The configuration information for fields that require parsing or reformatting and, optionally, normalization, before being processed by the standardization engine.]
[group]	[The configuration information for the national domain and the source and target fields for one standardization unit. You can define multiple group elements.]
[unstandardized-source-fields]	[A list of fields to be standardized.]
[unstandardized-source-field-name]	[A field to be standardized. If you define more than one source field in the same standardization unit, the fields are concatenated during standardization with a pipe (	) between lines (for the Master Index Standardization Engine).]

[Element/Attribute]	[Description]
[standardization-targets]	[A list of fields in which the standardized data from the source fields is stored.]
[target-mapping]	[The configuration information for one destination field in which standardized data from the source field will be stored. One source field will likely have several destination fields.]
[standardized-target-field-name]	[The ePath of the destination field in the object where the standardized value will be saved (for example, Person.Address[*].StreetName).]
[phoneticize-fields]	[A list of fields to be phonetically encoded.]
[phoneticize-field]	[The configuration information for each field to be phonetically encoded, including the encoder to use.]
[unphoneticized-source-field-name]	[The ePath of the source field in the system object from which the value to phonetically encode will be retrieved (for example, Person.Address[*].StreetName).] [Note: This can refer to the original field or to a standardized or normalized field.]
[phoneticized-target-field-name]	[The ePath of the field in which the phonetically encoded value will be saved in the system object.]
[phoneticized-object-field-id]	[A field ID to identify the field to the phonetic encoder. This is not currently used with the OHMPI Standardization Engine.]
[encoding-type]	[The phonetic encoder to use for this field. This must correspond to the encoding-type configured for the desired encoder in the PhoneticEncodersConfig element.]
[MatchingConfig]	[The configuration information for the match string (that is, the fields that are included in the data string sent to the match engine and against which weighting is performed). The attributes of the MatchingConfig element define the module name and Java class, and their default values should not be changed.]
[match-system-object]	[The configuration and field definitions for the match string.]
[object-name]	[The name of the object containing the fields in the match string. If you specify the parent object, you can specify fields from the parent and any child object in the match string.]
[match-columns]	[A list of fields in the match string. This element contains multiple match-column elements.]
[match-column]	[The configuration information for one field in the match string. You will use multiple match-column elements.]
[column-name]	[The fully qualified field name that defines the location of each field on which to match (for example, Enterprise.SystemSBR.Person.Address.City).]
[match-type]	[The type of matching performed on the specified field. This is an ID that is specific to the match engine and identifies the field to the match engine. This value must correspond to a match type defined for the match engine.]
[match-order]	[An integer specifying the order in which the field appears in the match string. This element is optional. If no order is specified, matching is performed in the order in which the fields are listed.]

[Element/Attribute]	[Description]
[MEFAConfig]	[The configuration information for the components of the matching service. The MEFAConfig attributes (module-name and parser-class) define the module name and Java class, and their default values should not be changed. You should only change the names of the component classes in this section if you created a corresponding custom component.]
[block-picker]	[The configuration information for the Java class that chooses which block of criteria defined for the blocking query to use for each match pass.]
[class-name]	[The name of the block picker Java class.]
[pass-controller]	[The configuration information for the Java class that determines whether the blocking query should continue performing match passes after each match pass is complete.]
[class-name]	[The name of the pass controller Java class.]
[standardizer-api]	[The configuration information for the standardization engine to use.]
[class-name]	[The name of the standardizer API Java class.]
[standardizer-config]	[The configuration information for the Java class that provides configuration information to the standardization engine.]
[class-name]	[The name of the standardizer configuration Java class.]
[matcher-api]	[The configuration information for the match engine to use.]
[class-name]	[The name of the match engine API Java class.]
[matcher-config]	[The configuration information for the Java class that provides configuration information to the match engine.]
[class-name]	[The name of the match engine configuration Java class.]
[PhoneticEncodersConfig]	[The configuration information for the phonetic encoders used by the master index application. The attributes (module-name and parser-class) define the module name and Java class. The default values should not be changed.]
[encoder]	[A list of phonetic encoders used by the standardization engine.]
[encoder-implementation-class]	[The fully qualified name of the Java class that determines the behavior of the phonetic encoder. The following default classes are defined for the OHMPI Match Engine.] ■ [[[com.sun.mdm.index.phonetic.impl.Nysiis]] ■ [[com.sun.mdm.index.phonetic.impl.Soundex]] ■ [[com.sun.mdm.index.phonetic.impl.Metaphone]] ■ [[com.sun.mdm.index.phonetic.impl.DoubleMetaphone]] ■ [[com.sun.mdm.index.phonetic.impl.RefinedSoundex]] ■ [[com.sun.mdm.index.phonetic.impl.SoundexFR]]]

[mefa.xml Example]

[Below is a short sample of mefa.xml based on a master person index application processing person data. This sample covers the basic elements of mefa.xml, but a production environment would contain several more fields to standardize as well as several additional match string fields.]

```
[[<StandardizationConfig module-name="Standardization" parser-class=]
["com.sun.mdm.index.configurator.impl.standardization.StandardizationConfiguration
```

```
">]
[   <standardize-system-object>]
[       <system-object-name>Person</system-object-name>]
[       <structures-to-normalize>]
[          <group standardization-type="PersonName" domain-selector=]
[            "com.sun.mdm.index.matching.impl.SingleDomainSelectorUS">]
[              <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>]
[                  <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>FirstName]
[                     </standardized-object-field-id>]
[                     <standardized-target-field-name>]
[                         Person.Alias[*].StdFirstName]
[                     </standardized-target-field-name>]
[                  </target-mapping>]
[                  <target-mapping>]
[                     <standardized-object-field-id>LastName]
[                     </standardized-object-field-id>]
[                     <standardized-target-field-name>]
[                         Person.Alias[*].StdLastName]
[                     </standardized-target-field-name>]
[                  </target-mapping>]
[              </normalization-targets>]
[          </group>]
[          <group standardization-type="PersonName" domain-selector=]
[            "com.sun.mdm.index.matching.impl.SingleDomainSelectorUS">]
[              <unnormalized-source-fields>]
[]
[                  <source-mapping>]
[                     <unnormalized-source-field-name>Person.FirstName]
[                     </unnormalized-source-field-name>]
[                     <standardized-object-field-id>FirstName]
[                     </standardized-object-field-id>]
[                  </source-mapping>]
[                  <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>FirstName]
[                     </standardized-object-field-id>]
```

```
[                        <standardized-target-field-name>Person.StdFirstName]
[                        </standardized-target-field-name>]
[                    </target-mapping>]
[                    <target-mapping>]
[                        <standardized-object-field-id>LastName]
[                        </standardized-object-field-id>]
[                        <standardized-target-field-name>Person.StdLastName]
[                        </standardized-target-field-name>]
[                    </target-mapping>]
[                </normalization-targets>]
[            </group>]
[        </structures-to-normalize>]
[        <free-form-texts-to-standardize>]
[            <group standardization-type="Address" domain-selector=]
[              "com.sun.mdm.index.matching.impl.MultiDomainSelector">]
[                <locale-field-name>Person.Country</locale-field-name>]
[                <locale-maps>]
[                    <locale-codes>]
[                        <value>Default</value>]
[                        <locale>US</locale>]
[                    </locale-codes>]
[                </locale-maps>]
[                <unstandardized-source-fields>]
[                    <unstandardized-source-field-name>]
[                     Person.Address[*].AddressLine1]
[                    </unstandardized-source-field-name>]
[                    <unstandardized-source-field-name>]
[                     Person.Address[*].AddressLine2]
[                    </unstandardized-source-field-name>]
[                </unstandardized-source-fields>]
[                <standardization-targets>]
[                    <target-mapping>]
[                        <standardized-object-field-id>HouseNumber]
[                        </standardized-object-field-id>]
[                        <standardized-target-field-name>]
[                         Person.Address[*].HouseNumber]
[                        </standardized-target-field-name>]
[                    </target-mapping>]
[                    <target-mapping>]
[                        <standardized-object-field-id>MatchStreetName]
[                        </standardized-object-field-id>]
[                        <standardized-target-field-name>]
[                         Person.Address[*].StreetName]
[                        </standardized-target-field-name>]
[                    </target-mapping>]
[                    <target-mapping>]
[                        <standardized-object-field-id>]
[                         StreetNamePrefDirection]
[                        </standardized-object-field-id>]
[                        <standardized-target-field-name>]
[                         Person.Address[*].StreetDir]
[                        </standardized-target-field-name>]
[                    </target-mapping>]
[                    <target-mapping>]
[                        <standardized-object-field-id>StreetNameSufType]
[                        </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>]
[        <phoneticize-fields>]
[            <phoneticize-field>]
[                <unphoneticized-source-field-name>Person.FirstName_Std]
[                </unphoneticized-source-field-name>]
[                <phoneticized-target-field-name>Person.FirstName_Phon]
[                </phoneticized-target-field-name>]
[                <encoding-type>Soundex</encoding-type>]
[            </phoneticize-field>]
[            <phoneticize-field>]
[                <unphoneticized-source-field-name>Person.LastName_Std]
[                </unphoneticized-source-field-name>]
[                <phoneticized-target-field-name>Person.LastName_Phon]
[                </phoneticized-target-field-name>]
[                <encoding-type>NYSIIS</encoding-type>]
[            </phoneticize-field>]
[            <phoneticize-field>]
[                <unphoneticized-source-field-name>]
[                 Person.Address[*].StreetName]
[                </unphoneticized-source-field-name>]
[                <phoneticized-target-field-name>]
[                 Person.Address[*].StreetNamePhoneticCode]
[                </phoneticized-target-field-name>]
[                <encoding-type>NYSIIS</encoding-type>]
[            </phoneticize-field>]
[        </phoneticize-fields>]
[    </standardize-system-object>]
[</StandardizationConfig>]
[<MatchingConfig module-name="Matching" parser-class=]
[ "com.sun.mdm.index.configurator.impl.matching.MatchingConfiguration">]
[    <match-system-object>]
[        <object-name>Person</object-name>]
[        <match-columns>]
[            <match-column>]
[              <column-name>Enterprise.SystemSBR.Person.StdFirstName]
[              </column-name>]
[              <match-type>FirstName</match-type>]
[            </match-column>]
[            <match-column>]
[              <column-name>Enterprise.SystemSBR.Person.StdLastName]
[              </column-name>]
[              <match-type>LastName</match-type>]
[            </match-column>]
[            <match-column>]
[              <column-name>Enterprise.SystemSBR.Person.DOB</column-name>]
[              <match-type>DOB</match-type>]
[            </match-column>]
[        </match-columns>]
[    </match-system-object>]
[</MatchingConfig>]
[<MEFAConfig module-name="MEFA" parser-class=]
[ "com.sun.mdm.index.configurator.impl.MEFAConfiguration">]
[    <block-picker>]
[      <class-name>com.sun.mdm.index.matching.impl.PickAllBlocksAtOnce]
[      </class-name>]
[    </block-picker>]
[    <pass-controller>]
[      <class-name>com.sun.mdm.index.matching.impl.PassAllBlocks]
```

```
[          </class-name>]
[    </pass-controller>]
[        <class-name>]
[         com.sun.mdm.index.matching.adapter.SbmeStandardizerAdapter]
[        </class-name>]
[    </standardizer-api>]
[    <standardizer-config>]
[        <class-name>]
[         com.sun.mdm.index.matching.adapter.SbmeStandardizerAdapterConfig]
[        </class-name>]
[    </standardizer-config>]
[    <matcher-api>]
[        <class-name>com.sun.mdm.index.matching.adapter.SbmeMatcherAdapter]
[        </class-name>]
[    </matcher-api>]
[    <matcher-config>]
[        <class-name>]
[         com.sun.mdm.index.matching.adapter.SbmeMatcherAdapterConfig]
[        </class-name>]
[    </matcher-config>]
[</MEFAConfig>]
[<PhoneticEncodersConfig module-name="PhoneticEncoders" parser-class=]
[ "com.sun.mdm.index.configurator.impl.PhoneticEncodersConfig">]
[    <encoder>]
[      <encoding-type>NYSIIS</encoding-type>]
[      <encoder-implementation-class>]
[       com.sun.mdm.index.phonetic.impl.Nysiis]
[      </encoder-implementation-class>]
[    </encoder>]
[    <encoder>]
[      <encoding-type>Soundex</encoding-type>]
[      <encoder-implementation-class>]
[       com.sun.mdm.index.phonetic.impl.Soundex]
[      </encoder-implementation-class>]
[    </encoder>]
[</PhoneticEncodersConfig>]
[]]]]]
```

[[Configuring Survivor Strategy]

[The Update Manager contains the logic used to generate the single best record (SBR) for a given object. The SBR is defined by a mapping of fields from external systems to the SBR, allowing you to define the fields from each system that are kept in the SBR. For each field in the SBR, an ePath denotes the location in the external system records from which the value is retrieved. Since there can be many external systems, you can optionally specify a strategy to select the SBR field from the list of external values. You can also specify any additional fields that might be required by the selection strategy to determine which external system contains the best data (by default, the record's update date and time is always taken into account). The Update Manager also specifies any custom Java classes to be used for different types of update transactions, such as merges, unmerges, changes to existing records, and new record inserts.]

[The Update Manager is configured in update.xml. The following topics describe the Update Manager and update.xml.]

- [[[["The Survivor Calculator and the SBR"]]]]

- [[[["Update Manager Components"]]]]

- [[[["The update.xml File"]]]]

[[The Survivor Calculator and the SBR]

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

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

> **[Note]:**   [Phonetic and standardized fields do not need to be defined in update.xml since their field values are determined by the standardization engine for the SBR.]

[[Update Manager Components]

[The logic that determines how the fields in the SBR are populated and how certain updates are performed is highly configurable in a master person index application, allowing you to design and develop the match strategy that best suits your processing requirements.]

[Configuring the Update Manager consists of customizing the following components:]

- [[[["Survivor Helper"]]]]
- [[["Weighted Calculator"]]]
- [[["Update Manager Policies"]]]]

[[Survivor Helper]

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

[For each field, you can specify system fields to be taken into consideration as well as a specific survivorship strategy. There are three basic strategies provided by Oracle Healthcare Master Person Index to determine survivorship for each field. You can define and implement custom strategies.]

- [[[Default Strategy]]
- [[Weighted Strategy]]
- [[Union Strategy]]]

[You can further configure the strategy for each field by filtering out unwanted or invalid values from the SBR. For more information, see ["Configuring SBR, Matching, and Blocking Filters" on page 2-53].]

[Survivor Helper Default Strategy]

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

[The default survivor strategy is com.sun.mdm.index.survivor.impl.DefaultSurvivorStrategy.]

[Survivor Helper Weighted Strategy]

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

[Survivor Helper Union Strategy]

[The weighted survivor strategy is com.sun.mdm.index.survivor.impl.WeightedSurvivorStrategy. You can define general weighted calculations to be performed by default for each field, and you can define specialized calculations to be performed for specific fields.]

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

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

[The union strategy is com.sun.mdm.index.survivor.impl.UnionSurvivorStrategy.]]

[[Weighted Calculator]

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

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

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

[The weighted calculator strategies include:]

- [[[SourceSystem]]
- [[SystemAgreement]]
- [[MostRecentModified]]]

[Weighted Calculator SourceSystem Strategy]

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

[Weighted Calculator SystemAgreement Strategy]

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

[Weighted Calculator MostRecentModified Strategy]

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

[[Update Manager Policies]

[The Update Manager policies specify custom Java classes that provide additional processing logic for each type of update transaction. By default, this additional processing is not defined in a standard master person index application. You can define custom update policies by creating the custom classes in the Source Packages node of the EJB project associated with the main master person index project. NetBeans also provides the ability to build and compile the custom Java code, and Oracle Healthcare Master Person Index automatically incorporates the classes when you generate the application. The Java classes defining the update policies are specified for the master person index application in the UpdateManagerConfig element of update.xml.]

[Update Manager Update Policies]

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

- [[[[**Enterprise Merge Policy**] - The enterprise merge policy defines additional processing to perform when two enterprise objects are merged. This policy is defined by the EnterpriseMergePolicy element.]]

- [[[**Enterprise Unmerge Policy**] - The enterprise unmerge policy defines additional processing to perform when an unmerge transaction occurs. This policy is defined by the EnterpriseUnmergePolicy element.]]

- [[[**Enterprise Update Policy**] - The enterprise update policy defines additional processing to perform when a record is updated. This policy is defined by the EnterpriseUpdatePolicy element.]]

- [[[**Enterprise Create Policy**] - The enterprise create policy defines additional processing to perform when a new record is inserted into the master person index database. This policy is defined by the EnterpriseCreatePolicy element.]]

- [[[**System Merge Policy**] - The system merge policy defines additional processing to perform when two system objects are merged. This policy is defined by the SystemMergePolicy element.]]

- [[[**System Unmerge Policy**] - The system unmerge policy defines additional processing to perform when system objects are unmerged. This policy is defined by the SystemUnmergePolicy element.]]

- [[[**UndoAssumeMatchPolicy**] - The undo assume match policy defines additional processing to perform when an assumed match transaction is reversed. This policy is defined by the UndoAssumeMatchPolicy element.]]]

[Update Manager Update Policy Flag]

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

[[The update.xml File]

[The properties for the update process are defined in update.xml. Some of the information entered into the default configuration file is based on the fields defined in the wizard and some is standard across all implementations. For most implementations, this file will require customization.]

[The following topics provide information about working with update.xml:]

- [[[["Modifying update.xml"]]]]
- [[["The update.xml File Structure"]]]]]

[[Modifying update.xml]

[You can customize the configuration of the Update Manager by modifying update.xml. This file cannot be modified using the Configuration Editor; you need to modify the file directly. You can modify this file at any time, but it is not recommended after moving into production. The configuration controls how the SBR for each object is created, and modifying the file can cause discrepancies in how SBRs are formed before and after the modifications. It might also cause discrepancies in match results, since matching is performed against the SBR. You must regenerate the application and redeploy the project after modifying this file. The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes.]]

[[The update.xml File Structure]

[This topic describes the structure of the XML file, general requirements, and constraints. It also provides a sample implementation.]

[update.xmlFile Description]

[Table 12 lists each element in update.xml and provides a description of each element along with any requirements or constraints for each element.]

[Table 12 update.xml File Structure]

[Element/Attribute]	[Description]
[SurvivorHelperConfig]	[The configuration of the overall survivor strategy. The SurvivorHelperConfig attributes (module-name and parser-class) define the module name and Java class, and their default values should not be changed.]
[helper-class]	[The Java class that determines how to retrieve values from system records and to set them in the SBR. The default class uses the ePath notation to retrieve and set the values.]
[default-survivor-strategy]	[The configuration information for the survivor strategy to use as the default. You can define multiple survivor strategies and use different strategy combinations for the candidate fields. Any field that is not assigned a specific survivor strategy in the candidate-definitions list uses the default survivor strategy specified here.]
[strategy-class]	[The Java class name of the default strategy.]
[parameters]	[A list of optional parameters for the default survivor strategy.]
[parameter]	[A parameter for the default survivor strategy. The default parameter points to another section in update.xml that configures the default class. There can be multiple parameter elements.]
[description]	[An optional element that briefly describes the parameter.]
[parameter-name]	[The name of the parameter.] ■ [[[For the DefaultSurvivorStrategy, this value is [**preferredSystem**].]] ■ [[For the WeightedSurvivorStrategy, this value is [**ConfigurationModuleName**].]] ■ [[For the UnionSurvivorStrategy. this is not used.]]]
[parameter-type]	[The Java data type for the parameter value. For both the DefaultSurvivorStrategy and the WeightedSurvivorStrategy, this value is java.lang.String.]
[parameter-value]	[The value of the named parameter.] ■ [[[For the DefaultSurvivorStrategy, this is the processing code of the source system from which the SBR field value is retrieved.]] ■ [[For the WeightedSurvivorStrategy, this is the name of the module-name element that defines the weighted calculator to use as the default strategy (by default, [**WeightedSurvivorCalculator**]).]]]
[candidate-definitions]	[The configuration information for the fields to be included in the SBR. For any field that does not use the default survivor strategy, an alternate strategy is defined. All fields that are included in the SBR must be listed here.]

[Element/Attribute]	[Description]
[candidate-field/name]	[The qualified field name for a field in the SBR (for more information about field notations, see [Chapter 3, "Master Person Index Field Notations"]).]
[description]	[A short description of the candidate field (this element is optional).]
[system-fields]	[A field (other than the candidate field) that is evaluated to determine the value for the SBR. One example of this would be to evaluate the last update date of the system records to determine which value is most recent. This element is not currently used by any of the standard survivor strategies provided with the master person index application, but might be useful when defining custom strategies.]
[field-name]	[The name of the field to use to determine the value for the SBR.]
[survivor-strategy]	[An alternate survivor strategy to use for the given field in place of the default strategy defined in the default-survivor-strategy element. If a strategy is not specifically defined for a field, the default strategy is used for that field.]
[strategy-class]	[The name of the Java class to use for the alternate survivor strategy.]
[WeightedCalculator]	[The configuration of the weighted calculator. By default, this is the strategy specified as the default strategy in the default-survivor-strategy element. The WeightedCalculator attributes (module-name and parser-class) define the module name and Java class, and their default values should not be changed unless you create a custom class.]
[default-parameters]	[The configuration information for the default weighted calculator logic for all fields except those whose logic is defined in the candidate-field element below.]
[parameter]	[A parameter for the default logic of the weighted calculator.]
[quality]	[The type of weighted calculation to perform. You can specify any of the following:] ■ [[[SourceSystem]]] ■ [[SystemAgreement]] ■ [[MostRecentModified]]] [For more information about these qualities, see [Weighted Calculator].]
[preference]	[The preferred value for the specified quality. This element is only used for the SourceSystem quality and must be a source system code.]
[utility]	[A value that indicates the reliability of the specified quality for determining the best field value for the SBR. You define the scale for the utility values.]
[candidate-field]	[A field for which you want to use custom logic for the weighted calculator. The logic you specify here overrides the logic defined in the default-parameters section, but only for the fields specified. Each candidate field is identified by a name attribute and defines the survivor strategies for one field.]
[candidate-field/name]	[The name of the candidate field for which you want to define override logic.]

[Element/Attribute]	[Description]
[parameter]	[A parameter configuring the weighted calculator for the candidate field. You can define multiple parameters for each candidate field.]
[quality]	[The type of weighted calculation to perform. You can specify any of the following:] ■ [[[SourceSystem]]] ■ [[SystemAgreement]] ■ [[MostRecentModified] [For more information about these qualities, see [Weighted Calculator].]]]
[preference]	[The preferred value for the specified quality. This element is only used for the SourceSystem quality and the preference must be a source system code.]
[utility]	[A value that indicates the reliability of the specified quality for determining the best field value for the SBR. You define the scale for the utility values.]
[UpdateManagerConfig]	[The configuration information for the Update Manager. This section defines a list of Java classes to manage custom processing for different types of transactions. You can create the custom classes in the Source Packages folder of the EJB project and then specify those classes here. The UpdateManagerConfig attributes (module-name and parser-class) define the module name and Java class, and their default values should not be changed.]
[EnterpriseMergePolicy]	[A class that defines additional processing to perform when two enterprise objects are merged.]
[EnterpriseUnmergePolicy]	[A class that defines additional processing to perform when two enterprise objects are unmerged.]
[EnterpriseUpdatePolicy]	[A class that defines additional processing to perform when a record is updated.]
[EnterpriseCreatePolicy]	[A class that defines additional processing to perform when a new record is created.]
[SystemMergePolicy]	[A class that defines additional processing to perform when two system objects are merged.]
[SystemUnmergePolicy]	[A class that defines additional processing to perform when system objects are unmerged.]
[UndoAssumeMatchPolicy]	[A class that defines additional processing to perform when an assumed match transaction is reversed.]
[SkipUpdateIfNoChange]	[An indicator of whether the update policies are carried out if no changes are made to the existing record. Specify "true" to prevent the update policies from being performed when no changes are made to an existing record.]

[update.xml Example]

[Below is a sample of update.xml using a very small object structure based on person data. Note that standardized and phonetic fields are included in the candidate fields to ensure that they are also included in the SBR. In this sample, all fields use the default strategy except those included in the Alias object, which uses the union strategy. The value that is populated in the LastName field of the SBR is dependent on the SSN field of the system objects. In addition, custom logic is defined only for the SSN field; the remaining fields use the default logic defined in the default-parameters element.]

```
[[<SurvivorHelperConfig module-name="SurvivorHelper" ]
[ parser-class="com.sun.mdm.index.configurator.impl.SurvivorHelperConfig">]
[    <helper-class>com.sun.mdm.index.survivor.impl.DefaultSurvivorHelper]
[    </helper-class>]
[    <default-survivor-strategy>]
[        <strategy-class>]
[         com.sun.mdm.index.survivor.impl.WeightedSurvivorStrategy]
[        </strategy-class>]
[        <parameters>]
[            <parameter>]
[                <parameter-name>ConfigurationModuleName</parameter-name>]
[                <parameter-type>java.lang.String</parameter-type>]
[                <parameter-value>WeightedSurvivorCalculator]
[                </parameter-value>]
[            </parameter>]
[        </parameters>]
[    </default-survivor-strategy>]
[    <candidate-definitions>]
[        <candidate-field name="Person.LastName">]
[            <system-fields>]
[                <field-name>Person.SSN</field-name>]
[            </system-fields>]
[        </candidate-field>]
[        <candidate-field name="Person.FirstName"/>]
[        <candidate-field name="Person.MiddleName"/>]
[        <candidate-field name="Person.DOB"/>]
[        <candidate-field name="Person.Gender"/>]
[        <candidate-field name="Person.SSN"/>]
[        <candidate-field name="Person.FnamePhoneticCode"/>]
[        <candidate-field name="Person.LnamePhoneticCode"/>]
[        <candidate-field name="Person.StdFirstName"/>]
[        <candidate-field name="Person.StdLastName"/>]
[        <candidate-field name="Person.Alias[*].*">]
[            <survivor-strategy>]
[                <strategy-class>]
[                 com.sun.mdm.index.survivor.impl.UnionSurvivorStrategy]
[                </strategy-class>]
[            </survivor-strategy>]
[        </candidate-field>]
[    </candidate-definitions>]
[</SurvivorHelperConfig>]
[<WeightedCalculator module-name="WeightedSurvivorCalculator"]
[ parser-class="com.sun.mdm.index.configurator.impl.WeightedCalculatorConfig">]
[    <candidate-field name="Person.SSN">]
[        <parameter>]
[            <quality>SourceSystem</quality>]
[            <preference>SBYN</preference>]
[            <utility>100.0</utility>]
[        </parameter>]
[        <parameter>]
[            <quality>MostRecentModified</quality>]
[            <utility>75.0</utility>]
[        </parameter>]
[    </candidate-field>]
[    <default-parameters>]
[        <parameter>]
[            <quality>MostRecentModified</quality>]
[            <utility>80.0</utility>]
[        </parameter>]
[        <parameter>]
```

```
[          <quality>SourceSystem</quality>]
[          <preference>SBYN</preference>]
[          <utility>100.0</utility>]
[      </parameter>]
[   </default-parameters>]
[</WeightedCalculator>]
[<UpdateManagerConfig module-name="UpdateManager" ]
[ parser-class="com.sun.mdm.index.configurator.impl.UpdateManagerConfig">]
[   <EnterpriseMergePolicy>com.sun.mdm.index.user.CustomMergePolicy]
[   </EnterpriseMergePolicy>]
[   <EnterpriseUnmergePolicy>com.sun.mdm.index.user.CustomUnmergePolicy]
[   </EnterpriseUnmergePolicy>]
[   <EnterpriseUpdatePolicy>com.sun.mdm.index.user.CustomUpdatePolicy]
[   </EnterpriseUpdatePolicy>]
[   <EnterpriseCreatePolicy>com.sun.mdm.index.user.CustomCreatePolicy]
[   </EnterpriseCreatePolicy>]
[   <SystemMergePolicy>com.sun.mdm.index.user.CustomSystemMergePolicy]
[   </SystemMergePolicy>]
[   <SystemUnmergePolicy>com.sun.mdm.index.user.CustomSystemUnmergePolicy]
[   </SystemUnmergePolicy>]
[   <UndoAssumeMatchPolicy>com.sun.mdm.index.user.CustomUndoMatchPolicy]
[   </UndoAssumeMatchPolicy>]
[   <SkipUpdateIfNoChange>true</SkipUpdateIfNoChange>]
[</UpdateManagerConfig>]
[]]
```

[Weighted Calculator Logic]

[The following sample illustrates how the weighted calculator uses the parameters you define to determine which field values to use in the SBR. Using this sample, if there is a value in only one of the system records but not in the other, that value is used in the SBR regardless of update date. If there is a value in both system records and they were updated at the same time, the SAP field value is used (80.0>30.0). If there is a value in both system records, but CDW was the most recently modified, the value from CDW is populated into the SBR ((30.0+70.0)>80.0)]

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

[[Configuring SBR, Matching, and Blocking Filters]

[In filter.xml, you can define values to be excluded during the SBR calculation, during the matching process, and during the blocking query. The following sections describe the structure of filter.xml and provide information about defining filters.]

- [[[["Master Person Index Field Filters"]]]
- [[["The filter.xml File"]]]]

[[Master Person Index Field Filters]

[Oracle Healthcare Master Person Index provides the ability to exclude unwanted values during key processes, such as blocking, matching, and SBR calculation. Data coming into a master person index application frequently contains default values that are used when the actual value is unknown. One of the most common examples is using "999-99-9999" or "000-00-0000" for a social security number. Another example is the occurrence in patient data when the name of a newborn baby is not yet known and the name is entered as "Baby", "Baby Boy", or "Baby Girl". Retrieving all of these values for a blocking query and performing subsequent matching on these values wastes valuable computer resources. Removing invalid or overused values from these key processes can improve the performance of the master person index application.]

[The following topics provide additional information about each type of filter:]

- [[[[SBR Filters]]]
- [[[Blocking Query Filters]]]
- [[[Match String Filters]]]
- [[[Exclusion Lists]]]]

[[SBR Filters]

[When the survivor calculator determines the values to populate in the SBR for a record, you want to eliminate any values that obviously do not represent the best value for the field. These are most likely default values that are used when the actual value of a field is unknown. When a filter is defined for a field and a system object contains an excluded value in that field, the survivor calculator ignores that value and uses a value from a different system record for the survivor calculator. If there is only one system record in the enterprise record and that system record contains an excluded value, the excluded value is used for the SBR since there is no other value to use.]

[As an example, if you define a SBR filter for FirstName to exclude the value "Baby" and an enterprise record contains two system records, one with a FirstName of "Baby" and one with a FirstName of "Joel", then the value populated into the SBR is "Joel" regardless of how the survivor calculator is defined. If you have the same filter definition with an enterprise record that contains only one system record and the value of the FirstName is "Baby", then the value populated into the SBR is "Baby".]]

[[Blocking Query Filters]

[When a message comes in to the master person index application, values from the message are used as criteria for the blocking query used for matching. Several queries are created depending on the number of blocks that are defined. If the incoming message contains common default values, the query could result in an inordinate number of possible matches being returned from the master person index database for the match process. You can reduce this overhead by excluding known invalid values from blocking query fields, thereby reducing the number of non-matching query results.]

[As an example, a blocking filter for the Phone field excludes the value "9999999999" and the blocking query contains a block on the FirstName and Phone fields. If an incoming record contains "9999999999" in the Phone field, the blocking query returns

no matching records for that specific block of the query. Note that records containing the excluded value might be returned by other blocks in the query that do not include the Phone field.]]

[[Match String Filters]

[When a master person index application matches incoming records against records that already exist in the master person index database, you want to be sure the composite weights are not artificially inflated due to matching on default values in certain fields. One of the most common problems in matching arises from the SSN (or other national identifier) in person data. This field should be one of the most reliable identifiers of a person since the number is unique to each person and the field is typically required so it should not be null. This means that if the SSN of a person is unknown, the person entering the data must enter some value that is not a valid SSN. Often the numbers "999999999" or "000000000" are used. If an incoming record contains one of these values, the match process returns the full agreement weight for the SSN field against other records containing the default data. We know this match value is meaningless in this case.]

[You can reduce the number of inaccurate matches and potential matches by defining an exclusion list for specific fields in the match string. When a match filter is defined against a field and an incoming record contains an excluded value, that value is ignored in the match process and does not contribute to the composite match weight.]]

[[Exclusion Lists]

[An exclusion list defines all values to filter out or ignore for a specific field. You can define exclusion lists directly in the filter.xml file or you can create exclusion lists in text files and reference those files from filter.xml. You should create an exclusion list file for each field for which filters are defined, and you might need to create separate files for a field whose excluded values for SBR processing do not match the excluded values for matching or blocking, for example.]]]

[[The filter.xml File]

[The filter.xml file provides a template from which you can define filters for the SBR, blocking query, or match process. The default version of the file does not define any exclusions, so you do not need to modify the file if you do not use the filter capability.]

[The following sections provide information about the filter.xml file.]

- [[[["Modifying filter.xml"]]]]
- [[[["filter.xml File Structure"]]]]
- [[[["filter.xml Example"]]]]]

[[Modifying filter.xml]

[You can modify filter.xml using the XML editor. The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes. When you modify this file, you must regenerate the application and redeploy the project for the changes to take effect.]]

[[filter.xml File Structure]

[filter.xml consists primarily of a list of fields, each with their own filter definitions. Each field is defined within a field element and the filters are defined within a value element. The following table describes the elements and attributes of filter.xml.]

[Element]	
[Description]	**[Attribute]**
[field]	[A filter definition for one field. The definition includes the following elements and attributes. You can define multiple filter definitions, and each can define filters for the SBR, blocking, matching, or any combination of the three.]
[sbr]	[An indicator of whether to apply the filter to the SBR. Specify [**true**] to apply the filter to the SBR; otherwise specify [**false**].]
[matching]	[An indicator of whether to apply the filter to the blocking query. Specify [**true**] to apply the filter to the blocking query; otherwise specify [**false**].]
[blocking]	[An indicator of whether to apply the filter to the matching process. Specify [**true**] to apply the filter to the matching process; otherwise specify [**false**].]
[name]	[The qualified name for the field; for example, Person.SSN or Person.Address.PostalCode. For more information about qualified field names, see [Qualified Field Name Notation].]
[value]	[A list of field-value elements that specify the values to filter.]
[field-value]	[A value to filter from the SBR, blocking query, or matching process. You can define multiple field values. To use values listed in a flat file, define a file element instead of a field-value element.]
[file]	[A definition of the file that contains the list of values to filter.]
[delimiter]	[The character that delimits the values listed in the exclusion list flat file.]
[file-name]	[The path and name of a file that contains the list of values to filter. Be sure the values in this file are delimited by the character specified above.]

[[filter.xml Example]

[The following example defines a filter for the SSN field for the SBR only, filtering out the values "999-99-9999" and "000-00-0000". When the survivor calculator determines that the field value for the SBR should be "999-99-9999" or "000-00-0000", the survivor calculator ignores that value and either chooses a different value or ignores the field altogether, depending on how survivorship is defined.]

```
[[<field sbr="true" matching="false" blocking="false">]
[  <name>Person.SSN</name>]
[  <value>]
[    <field-value>"999-99-9999"</field-value>]
[    <field-value>"000-00-0000"</field-value>]
[  </value>]
[</field>]
[]]
```

[The following example defines an exclusion list for matching and blocking, but not for the SBR. When a blocking query executes a query block that includes the DOB, it checks the values in the exclusion list and ignores any records where the DOB matches one of the values. When match weights are being generated, DOB fields that contain values found in the exclusion list are ignored.]

```
[[<field sbr="false" matching="true" blocking="true">]
[  <name>Person.DOB</name>]
[  <value>]
[    <file delimter=";">]
[      <file-name>"./filters/DOB.txt"</file-name>]
```

```
[    </file>]
[  </value>]
[</field>]
[]]
```
[The exclusion list file for the above example would look similar to the following:]

```
[[0000000000;2222222222;3333333333;...]
[]]]]]
```

[[Configuring Field Validation]

[You can define custom logic for field validations and then specify them in validation.xml to associate the logic with the master person index application. The custom logic is created as a Java class by defining custom Java classes in the Source Package folder of the EJB project. The custom validation classes must implement com.sun.mdm.index.objects.validation.ObjectValidator. The exception thrown is com.sun.mdm.index.objects.validation.exception.ValidationException.]

[The following section describe the structure of validation.xml and provide information about custom field validators.]

- [[[["The validation.xml File"]]]]

[[The validation.xml File]

[By default, validation.xml defines one validation rule named validate-local-id. This rule defines certain validations that are performed against local ID and system fields before they are entered into the database. The local ID validator verifies that the system code is valid, the local ID format is correct, the local ID is the correct length, and that neither field is null.]

[The following sections provide information about working with the validation.xml:]

- [[[["Modifying validation.xml"]]]
- [[["validation.xml File Structure"]]]
- [[["update.xml Example"]]]]

[[Modifying validation.xml]

[You can modify validation.xml using the XML editor. The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes. When you modify this file, you must regenerate the application and redeploy the project for the changes to take effect.]]

[[validation.xml File Structure]

[validation.xml consists primarily of a list of rules. Each rule is defined within the ValidationConfig element and is defined by attributes within a rules element. ["update.xml Example"] describes the elements and attributes of validation.xml.]

[Table 13 update.xml File Elements]

[Element or Attribute]	[Description]
[rules]	[The configuration information for the validation rules.]
[rule]	[The configuration information for a specific validation rule in a rules list.]

[Element or Attribute]	[Description]
[rule/name]	[A rule attribute that specifies a name for the validation rule.]
[rule/object-name]	[A rule attribute that specifies the name of the class that defines the object to which the validation rule is applied, such as SystemObject or ParentNameObject (where ParentName is the name of the parent object in the Object Definition).]
[rule/class]	[A rule attribute that specifies the complete path of the Java class containing the validation rule.]

[[update.xml Example]

[Plug the custom validation classes you create into the master person index application by specifying the name of the custom plug-in for the class in validation.xml, as shown below.]

```
[[<ValidationConfig module-name="Validation" ]
[parser-class=
"com.sun.mdm.index.configurator.impl.validation.ValidationConfiguration"]
[   <rules>]
[       <rule name="validate-auxiliary-id" object-name="PersonObject"]
[        class="com.sun.mdm.index.user.AuxiliaryId"/>]
[       <rule name="validate-birth-date" object-name="PersonObject"]
[        class="com.sun.mdm.index.user.BirthDate"/>]
[   </rules>]
[</ValidationConfig>]
[]]]]]
```

[[Configuring Master Index Data Manager]

[The Master Index Data Manager (MIDM) is the web-based user interface for the master person index application that allows you to monitor and modify data in the index. This interface is highly configurable, and can be customized by modifying midm.xml in the master person index project.]

[The following topics describe the MIDM and the midm.xml structure, and provide a sample of the file structure.]

- [[[["About the MIDM"]]]]
- [[[["MIDM Configuration Components"]]]]
- [[[["The midm.xml File Structure"]]]]]

[[About the MIDM]

[The MIDM is a web-based interface that allows you to manage and monitor the data in your master person index database. Using the MIDM, you can search for records; add, update, deactivate, and reactivate records; review and resolve potential duplicate records; compare records; and merge and unmerge records. You can also view a transaction history for each record, view an audit log of access to the database, and run reports on the state of the data and the transactions that have been performed. This interface is configurable, allowing you to customize certain processing properties as well as the appearance of the windows.]

[The MIDM facilitates the use of screen readers and other assistive technology by providing information through HTML tags. It also provides tooltips when the cursor is placed over links and images on the MIDM pages.]]

[[MIDM Configuration Components]

[You can configure several properties of the MIDM to display the information you want in the way you want, to define the way searches can be processed, and to define the criteria that can be used for each search. Certain implementation options are also configured in midm.xml, such as application server information, debug options, and security information.]

[The configurable properties of the user interface fall into these categories:]

- [[[["Object and Field Properties"]]]
- [[["Relationship Properties"]]]
- [[["Display Properties"]]]
- [[["Implementation Configuration"]]]]

[[Object and Field Properties]

[In midm.xml, you specify which objects appear on the MIDM windows and the order in which they appear. You can also specify the fields displayed in each object and configure properties for each field. This file controls the configuration of a field's name, length, order of appearance on the MIDM, required data type, whether text can be entered into a field or if it must be selected from a predefined value, whether a field or combination of fields must be unique to a parent or child object, and whether the value of the field is hidden under certain circumstances. You can also specify whether the format of a field is dependent on the value of a related field (for example, the format of a credit card number field could be dependent on the type of credit card specified).]]

[[Relationship Properties]

[In midm.xml, relationships define the hierarchy of the object types listed on the MIDM. By specifying relationships, you define parent and child nodes. The parent and child nodes you specify in the relationships element must also be defined in the node elements of the file. You can specify one parent object; the remaining objects must be child objects to the parent you define. The relationships section is dependent on the relationships section in object.xml, and should only be changed if corresponding changes are made to object.xml.]]

[[Display Properties]

[You can configure these display options for the MIDM: the appearance of pages, audit log availability, local ID field labels, and the configuration of the search pages.]

[Page Display Properties]

[You can configure several display properties for the pages that appear on the MIDM. For example, you can specify whether certain system fields are visible, the type of object to display on a page, and the name of the tabbed headings. You can also rearrange the order of pages or sub-pages.]

[Audit Log]

[In the display configuration, you can specify whether an audit log is maintained of all instances in which object information was accessed from the MIDM. If the log is maintained, then information about each instance of access can be viewed on the MIDM. This is especially useful in healthcare implementations, where privacy of information is mandated.]

[Local ID Labels]

[A local ID is a unique identification code assigned to a record by the system in which the record originated. By default, the fields that display the local IDs are named "Local ID" on the MIDM pages. This name can be modified to a name more recognizable by MIDM users.]

[Search Page Configuration]

[Of the configurable pages, the pages that might require the most configuration are the search pages. In addition to defining the number of records to display in the search results list, you can also specify the search criteria that appear and the types of searches allowed from the MIDM.]

[You can define and name several search pages for each primary MIDM window, each with their own configuration. For each search page, you specify groups of fields that are displayed in boxed areas. Each boxed area can represent a different type of search, such as a demographic search, address search, EUID search, and so on. For the Records Details page searches, you must also specify the search types available for each search you define, such as alphanumeric or phonetic. You can configure searches by specifying a name for the search, the maximum number of records to return, whether the results are weighted, and whether wildcard characters can be used.]

[When you define the search types for the Records Details page of the MIDM, you must specify a query for each type you define. The queries you specify must already be defined in query.xml.]]

[[Implementation Configuration]

[The midm.xml file defines certain information about the application server for the master person index implementation, such as the names of certain validation and management components, and debug parameters. Most of the implementation information is predefined.]]]

[[The midm.xml File Structure]

[MIDM properties are defined in midm.xml. Some of the information entered into the default configuration file is based on the fields you defined in the wizard, and some is standard across all implementations. For most implementations, this file will require customization.]

[The following sections provide information about working with midm.xml:]

- [[[["Modifying midm.xml"]]]]
- [[["midm.xml File Description"]]]
- [[["midm.xml File Example"]]]]

[[Modifying midm.xml]

[You can modify midm.xml at any time, but you must regenerate the application and redeploy the project after making any changes to the file. Changes made to this file do not affect match processing. The possible modifications to this file are restricted by the schema definition, so be sure to validate the file after making any changes. Certain properties of this file can be modified using the Configuration Editor, such as whether a field appears on a search page and is required for that search. Most MIDM properties need to be modified directly in this file.]]

[[midm.xml File Description]

[The following table lists each element in midm.xml and provides a description of each element along with any requirements or constraints. Note that not all elements can be used on all predefined pages.]

[Table 14 midm.xml File Structure]

[Element/Attribute]	[Description]
[[**Object and Field Properties**]]	[]
[node]	[A container element for a list of fields for one object along with their configuration information. Each object that appears on the MIDM must be defined in a node element. Most of the information you need to specify in the node elements is generated by the wizard, but you can modify the information as needed.]
	[Note: All fields defined in midm.xml must also be defined in object.xml; however, not all fields defined in object.xml need to be defined for the MIDM. Any fields or objects not listed in midm.xml will not appear on the MIDM.]
[name]	[The name of the object defined by the node.]
[display-order]	[The order in which the child object types appear in the enterprise records on the MIDM pages.]
[field]	[A container element defining the configuration information for a field on the MIDM. Each field that appears on the MIDM must be defined in a field element. A field element can contain the eleven configuration elements listed below.]
[name]	[The name of the field as it appears in the object definition file.]
[display-name]	[The name of the field as it appears on the MIDM.]
[display-order]	[The order in which the field appears on the MIDM. For example, specify [**1**] to indicate this is the first field on the MIDM pages, [**2**] to indicate it is the second field, and so on.]
[max-length]	[The maximum number of characters displayed on the MIDM for the field. If the field uses an input mask, make sure the length of the input mask matches the length specified here.]
[gui-type]	[The type of display for the field. Specify one of the following options.]
	■ [[[[**TextBox**] - A standard data entry field.]]
	■ [[[**MenuList**] - A field that must be populated by selecting from a drop-down list.]]
	■ [[[**TextArea**] - A long field that requires a scrollbar, such as a comments field.]]]
[value-type]	[The master person index data type for the data populated in the field. The following data types are supported:]
	■ [[[[**string**] - Contains a string of characters.]]
	■ [[[**date**] - Contains a date value.]]
	■ [[[**float**] - Contains a floating point integer.]]
	■ [[[**int**] - Contains an integer.]]
	■ [[[**char**] - Contains a single character.]]
	■ [[[**boolean**] - Contains either true or false.]]]

[Element/Attribute]	[Description]
[input-mask]	[A mask used by the MIDM to add punctuation to a field. You can add an input mask to display telephone numbers as "(123)456-7890" even though the database might store them as "1234567890" and the user enters the numbers with no punctuation. The following character types are allowed:]
	■ [[[[**D**] - Indicates a numeric character.]]
	■ [[[**L**] - Indicates an alphabetic character.]]
	■ [[[**A**] - Indicates an alphanumeric character.]
	[For example, the input mask for the above telephone format is "(DDD)DDD-DDDD".]
	[Note: If the length of the input mask is greater than the value specified for the max-length element, update the max-length property to match.]]]
[value-mask]	[A mask used by the master person index application to strip any extra characters that were added by the input mask to ensure that data is stored in the database in the correct format. This mask must be the same length as the input mask.]
	[To specify a value mask, type the same value as is entered for the input mask, but type an [**x**] in place of each punctuation mark. For example, using the above phone number example, you need to specify a value mask of xDDDxDDDxDDDD. A value mask is not required for date fields.]
[value-list]	[The name of the menu list used to populate the drop-down list for the field. This is required if the gui-type specified is MenuList, and it must match a code of an element in the sbyn_common_header database table.]
[is-sensitive]	[An indicator of whether the value of the field is hidden on the MIDM for records with a certain VIP status. Only users with the Administrator or Field_VIP user roles can view the hidden information. Specify [**true**] to hide the field value; specify [**false**] (or remove the is-sensitive element) to display the field value.]
	[Note: This element is only used if the object-sensitive-plug-in-class in the impl-details section is populated. It must precede the key-type element described below.]
[key-type]	[An indicator of whether the field (or a combination of key fields) must be unique in an enterprise record. Unique key fields identify unique child objects in an enterprise object. Specify [**true**] to indicate the field is a key field; specify [**false**] if it is not.]
[[**Relationship Properties**]]	[]
[relationships]	[A container element that defines the hierarchy of the parent and child objects displayed on the MIDM. This element contains the following two elements.]
[name]	[The name of the parent object.]
[children]	[The name of a child object. There should be one children element for each child object defined in the previous section of the file.]
[[**Implementation Properties**]]	[]

[Element/Attribute]	[Description]
[impl-details]	[A container element that defines the configuration information for certain classes that are required for the MIDM to connect to the server. This section also defines debug and security options.]
[master-controller-jndi-name]	[The JNDI name for the Master Controller. The default name is ejb/app-nameMasterController, where app-name is the name of the master person index application.]
[validation-service-jndi-name]	[The JNDI name for the processing code validator. The default name is ejb/app-nameCodeLookup.]
[usercode-jndi-name]	[The JNDI name for the user code validator (used for non-unique IDs). The default name is ejb/app-nameUserCodeLookup.]
[report-generator-jndi-name]	[The JNDI name for the MIDM report generator. The default name is ejb/app-nameReportGenerator.]
[debug-flag]	[An indicator of whether debug information is logged. Specify [**true**] to log debug information.]
[debug-dest]	[The destination to which debug information is written. Specify [**console**] to log debug information to a monitor; specify [**file**] to print to a file.]
[object-sensitive-plug-in-class]	[The name of the class that contains logic for masking the data in certain fields from certain users. For example, certain sensitive information should only be viewed by administrators. If you specify field masking, you must define a custom plug-in to handle the process and specify it here.]
[[**MIDM Page Properties**]]	[]
[gui-definition]	[A container element that defines the configuration information for the pages that appear on the MIDM, including the types of searches available and any sub-screens contained on those pages.]
[page-definition]	[A container element for the configuration elements that define the individual MIDM pages.]
[initial-screen-id]	[The screen ID of the first page to appear when you log in to the MIDM. Enter the value defined in the screen-id element of the page you want to display (described later in this table).]
[local-id]	[The name to use for all fields and columns containing local IDs. If you want to change the field label from "Local ID" to a name that is more relevant to your implementation, define that name here. In fields where the local ID label is abbreviated to "LID1" or "LID2", the name becomes local-id1 or local-id2 (where local-id is the value specified for this element). This name is also perpetuated to heading labels on search pages.]

[Element/Attribute]	[Description]
[page-name]	[A name that identifies the page to the MIDM. Each page is defined by a page-name element that contains all of the configuration information for the page. The name of this element is specific to the page being defined, and the allowed values for the default MIDM include the following:]
	■ [[[dashboard]]
	■ [[record-details]]
	■ [[transactions]]
	■ [[duplicate-records]]
	■ [[assumed-matches]]
	■ [[source-record]]
	■ [[reports]]
	■ [[audit-log]]]
[allow-insert]	[An indicator of whether entries are written to the audit log each time data is accessed on the MIDM. This element is only used when defining the Audit Log page. Specify [**true**] to enable the audit log. Specify [**false**] to disable the audit log. If the audit log is maintained, the Audit Log page is available on the MIDM for searching and viewing audit log entries and the information is stored in the sbyn_audit table.]
[root-object]	[The name of the parent object. This should not be changed.]
[tab-name]	[A name for tabbed heading for the page. This name appears on tab labels associated with the pages on the MIDM.]
[screen-id]	[A unique identifier for the page. This is referenced from the initial-screen-id element to specify the initial page that appears when a user logs in to the MIDM. This value must be an integer.]
[display-order]	[The order in which the page appears in the tabbed headings. The display order goes from left to right.]
[search-pages]	[A container element that lists and defines the searches that are available from the page. You can define multiple searches for each page.]
[[**Search Page Properties**]]	[]
[simple-search-page]	[A container element that defines one type of search for an MIDM page. You can define multiple simple search pages.]
[screen-title]	[The name of the search as it appears on the search page. Users can select a type of search to perform based on the titles you define for the searches here.]
[search-result-id]	[The unique identifier for the search results page that appears for the search. You can define multiple search results pages for each MIDM page, and each results page has a unique ID. You can enter any of the search result ID values defined in the search-results-pages element described later in this table.]
[search-screen-order]	[The order in which the search page appears on the MIDM. The display order goes from left to right.]
[show-euid]	[An indicator of whether to display the EUID field in the search criteria. Specify [**true**] to display the EUID; otherwise specify [**false**]. This element is not used by all pages, and the default value is false if this element does not exist.]

[Element/Attribute]	[Description]
[show-lid]	[An indicator of whether to display the local ID and system fields in the search criteria. Specify [**true**] to display the fields; otherwise specify [**false**]. This element is not used by all pages, and the default value is false if this element does not exist. When the local ID is displayed, the local ID and system fields appear in their own labelled box.]
[show-status]	[An indicator of whether to display the record status field in the search results list. Specify [**true**] to display the field; otherwise specify [**false**]. This element is not used by all pages, and the default value is false if this element does not exist.]
[show-create-date]	[An indicator of whether to display the create date field to allow searching on the date the record was created. Specify [**true**] to display the field; otherwise specify [**false**]. This element is not used by all pages, and the default value is false if this element does not exist.]
[show-create-time]	[An indicator of whether to display the create time field to allow searching on the time the record was created. Specify [**true**] to display the field; otherwise specify [**false**]. This element is not used by all pages, and the default value is false if this element does not exist.]
[show-timestamp]	[An indicator of whether to display the timestamp field. Specify [**true**] to display the field; otherwise specify [**false**]. This element is not used by all pages, and the default value is false if this element does not exist.]
[instruction]	[A short statement to help the user process a search. The text you enter here appears above the search fields on the Search page.]
[field-group]	[A list of fields that appear on the Search page. You can define multiple field groups, and each group is contained in a labelled box on the Search page. Note that you can only define fields for searches on the Record Details page.]
[description]	[A description of the fields defined for the field-group element. This value appears as a box label for the area of the page that contains the specified fields.]
[field-ref]	[One field definition for a field in the field group. Use the simple field name of the field with their corresponding objects as the root. For example, the path to the FirstName field in the Person object is "Person.FirstName". You can define multiple field-ref elements for each field group, each of which are further configured by the following two optional attributes.]
[field-ref/required]	[An indicator of whether the field is required in order to perform a search. Specify any of the following values.] ■ [[[[**true**] - The corresponding field is required to perform the search. These fields are marked with an asterisk (*).]] ■ [[[**false**] - The corresponding field is not required to perform the search. If the required attribute is not defined, the default is false.]] ■ [[[**oneof**] - This is assigned to more than one field and at least one of the fields with this designation is required to perform the search. If a group of fields is designated as "oneof", those fields are marked with a dagger () on the search page.]]]

[Element/Attribute]	[Description]
[field-ref/choice]	[An indicator of whether you search by a range of values rather than an exact value for the field. Specify one of the following values.]
	■ [[[[exact] - The search is performed on the exact value entered (wildcard may be allowed). If the choice attribute is not specified, this is the default value.]]
	■ [[[range] - The search is performed on a range of values based on the entered search criteria. Fields with this designation appear twice on the search page, once with "From" appended to the field label and once with "To" appended to the field label. Be sure any searches that use range searching in the MIDM are configured to use range searching in query.xml.]]]
	[To define a field for both exact and range searching, define the field twice; once with this attribute set to exact and once with it set to range.]
[search-option]	[A container element for the configuration information for a search. Each search-option element defines one type of search for the page. Note that not all search pages use search options.]
[display-name]	[A short phrase describing the type of search to perform, such as "Alphanumeric Search" or "Phonetic Search". These names appear on the MIDM for users to select when specifying the type of search to perform.]
[query-builder]	[The type of query to use when this type of search is selected. The value entered here must match a query-builder name in query.xml.]
[weighted]	[An indicator of whether the results of the search are assigned matching probability weights. Specify [**true**] to assign matching weights or [**false**] to return unweighted results.]
[candidate-threshold]	[The maximum number of records to return for a search. This value must be a positive number, and is only used for blocking queries. Setting the candidate threshold to zero is equivalent to not setting a threshold.]
[parameter]	[A list of optional parameters for the search. Not all query builders require parameters.]
[name]	[The name of the parameter. Currently, only UseWildCard is available.]
[value]	[The value of the parameter. For the UseWildCard parameter, this is an indicator of whether the parameter is enabled or disabled. Specify [**true**] to allow wildcard characters or [**false**] to perform exact-match searches.]
[[**Search Results List Properties**]]	[]
[search-result-pages]	[A container element for a list of search result page definitions. This allows you to have more than one search result configuration for the same MIDM page.]
[search-result-list-page]	[A container element for the configuration information for one search results page. You can define multiple search results pages.]
[search-result-id]	[A unique identifier for the search result page. This element identifies the results page to display for each search type defined above. This value must be an integer.]
[item-per-page]	[The number of resulting records to display on one page.]

[Element/Attribute]	[Description]
[max-result-size]	[The maximum number of records to return for a search.]
[show-status]	[An indicator of whether to display the record status field in the results list. Specify [**true**] to display the field; otherwise specify [**false**]. Note that you cannot configure the search results fields for all search results lists.]
[show-create-date]	[An indicator of whether to display the create date field in the results list. Specify [**true**] to display the field; otherwise specify [**false**].]
[show-create-time]	[An indicator of whether to display the create time field in the results list. Specify [**true**] to display the field; otherwise specify [**false**].]
[show-timestamp]	[An indicator of whether to display the timestamp field in the results list. Specify [**true**] to display the field; otherwise specify [**false**].]
[field-group]	[A container element for one list of fields that appear in the search results. You can define multiple groups of fields.]
[description]	[A brief description for the field group. This value appears on the MIDM above the fields that are returned from a search.]
[field-ref]	[A definition for one field that appears in the search results list. Use the simple field names with their corresponding objects as the root. For example, the path to the FirstName field in the Person object is "Person.FirstName". You can define multiple field-ref elements.]
[[**Subscreen Properties**]]	[]
[subscreen-configurations]	[A list of sub-pages for a page-name element, described earlier in this table. You can define several pages within the main tabbed page. These pages appear on the page for which they are defined with their own tabbed headings beneath the main tabbed heading row.]
[subscreen]	[A container element for a sub-page. This element uses the same sequence of elements as the main page-name element, including the following:] ■ [[[enable]] ■ [[root-object]] ■ [[tab-name]] ■ [[report-name]] ■ [[screen-id]] ■ [[display-order]] ■ [[search-pages]] ■ [[search-result-pages]]] [The subscreen element does not include the allow-insert element, but does include two new elements, enable and report-name. The enable element contains a Boolean indicator of whether the tabbed sub-page is available. The report-name element is used for defining reports.]

[Note]:   [The subscreen element is used by default to define standard reports that can be run from the MIDM. In these subscreen definitions, the report-name element indicates the type of report being generated.]

[You can specify any of the following production reports:]

- [[[Assumed Matches]]

- [[Potential Duplicate]]

- [[Deactivated]]

- [[Merged]]

- [[Unmerged]]

- [[Update]]]

[Or you can specify any of the following activity reports.]

- [[[Weekly Transaction Summary Report]]

- [[Monthly Transaction Summary Report]]

- [[Yearly Transaction Summary Report]]]]

[[midm.xml File Example]

[Below is a short excerpt from midm.xml based on a master person index application processing person information. This sample defines two pages on the MIDM. The first page defines one blocking search, one simple lookup search, and one search result list. The second page is the reports page, and it defines a Potential Duplicate Report and a Weekly Activity report.]

```
[[[<node>]
[    <name>Person</name> ]
[    <field>]
[        <name>LastName</name> ]
[        <display-name>Last Name</display-name>]
[        <display-order>1</display-order>]
[        <max-length>40</max-length>]
[        <gui-type>TextBox</gui-type>]
[        <value-type>string</value-type>]
[        <key-type>true</key-type>]
[    </field>]
[    <field>]
[        <name>FirstName</name> ]
[        <display-name>First Name</display-name>]
[        <display-order>2</display-order>]
[        <max-length>40</max-length>]
[        <gui-type>TextBox</gui-type>]
[        <value-type>string</value-type>]
[        <key-type>true</key-type>]
[    </field>]
[    <field>]
[        <name>DOB</name> ]
[        <display-name>DOB</display-name>]
[        <display-order>3</display-order>]
[        <max-length>32</max-length>]
[        <gui-type>TextBox</gui-type>]
[        <value-type>date</value-type>]
[        <key-type>true</key-type>]
[    </field>]
[    <field>]
[        <name>Gender</name> ]
[        <display-name>Gender</display-name>]
[        <display-order>4</display-order>]
[        <max-length>8</max-length>]
```

```
[        <gui-type>MenuList</gui-type>]
[        <value-list>GENDER</value-list>]
[        <value-type>string</value-type>]
[        <key-type>true</key-type>]
[    </field>]
[    <field>]
[        <name>SSN</name> ]
[        <display-name>SSN</display-name>]
[        <display-order>5</display-order>]
[        <max-length>16</max-length>]
[        <gui-type>TextBox</gui-type>]
[        <value-type>string</value-type>]
[        <input-mask>DDD-DD-DDDD</input-mask>]
[        <value-mask>DDDxDDxDDDD</value-mask>]
[        <is-sensitive>true</is-sensitive>]
[    <field>]
[</node>]
[<node>]
[    <name>Alias</name>]
[    <display-order>1</display-order>]
[    <field>]
[        <name>LastName</name> ]
[        <display-name>LastName</display-name>]
[        <display-order>1</display-order>]
[        <max-length>40</max-length>]
[        <gui-type>TextBox</gui-type>]
[        <value-type>string</value-type>]
[        <key-type>true</key-type>]
[    </field>]
[    <field>]
[        <name>FirstName</name> ]
[        <display-name>FirstName</display-name>]
[        <display-order>2</display-order>]
[        <max-length>40</max-length>]
[        <gui-type>TextBox</gui-type>]
[        <value-type>string</value-type>]
[        <key-type>true</key-type>]
[    </field>]
[</node>]
[<relationships>]
[    <name>Person</name>]
[    <children>Alias</children>]
[</relationships>]
[<impl-details>]
[    <master-controller-jndi-name>ejb/PersonMasterController]
[    </master-controller-jndi-name>]
[    <validation-service-jndi-name>ejb/PersonCodeLookup]
[    </validation-service-jndi-name>]
[    <usercode-jndi-name>ejb/PersonUserCodeLookup</usercode-jndi-name>]
[    <reportgenerator-jndi-name>ejb/PersonReportGenerator]
[    </reportgenerator-jndi-name>]
[    <debug-flag>true</debug-flag>]
[    <debug-dest>console</debug-dest>]
[    <enable-security>true</enable-security>]
[    <object-sensitive-plug-in-class>com.sun.mdm.index.security.VIPPlugIn]
[    </object-sensitive-plug-in-class>]
[</impl-details>]
[<gui-definition>]
[    <page-definition>]
[        <local-id/>]
```

```
[          <initial-screen-id>1</initial-screen-id>]
[          <record-details>]
[              <root-object>Person</root-object>]
[              <tab-name>Record Details</tab-name>]
[              <screen-id>1</screen-id>]
[              <display-order>2</display-order>]
[              <search-pages>]
[                  <simple-search-page>]
[                      <screen-title>Advanced Person Lookup (Phonetic)</screen-title>]
[                      <search-result-id>1</search-result-id>]
[                      <search-screen-order>1</search-screen-order>]
[                      <show-euid>false</show-euid>]
[                      <show-lid>false</show-lid>]
[                      <instruction/>]
[                      <field-group>]
[                          <description>Person</description>]
[                          <field-ref required="false">Person.FirstName</field-ref>]
[                          <field-ref required="false">Person.LastName</field-ref>]
[                          <field-ref required="false">Person.SSN</field-ref>]
[                      </field-group>]
[                      <field-group>]
[                          <description>Alias</description>]
[                          <field-ref required="false">Person.Alias.FirstName</field-ref>]
[                          <field-ref required="false">Person.Alias.LastName</field-ref>]
[                      </field-group>]
[                      <search-option>]
[                          <display-name>Phonetic Search</display-name>]
[                          <query-builder>BLOCKER-SEARCH</query-builder>]
[                          <weighted>true</weighted>]
[                          <parameter>]
[                              <name>UseWildcard</name>]
[                              <value>false</value>]
[                          </parameter>]
[                      </search-option>]
[                  </simple-search-page>]
[                  <simple-search-page>]
[                      <screen-title>Simple Person Lookup</screen-title>]
[                      <search-result-id>1</search-result-id>]
[                      <search-screen-order>2</search-screen-order>]
[                      <show-euid>true</show-euid>]
[                      <show-lid>true</show-lid>]
[                      <instruction/>]
[                      <field-group/>]
[                      <search-option>]
[                          <display-name>Alpha Search</display-name>]
[                          <query-builder>ALPHA-SEARCH</query-builder>]
[                          <weighted>false</weighted>]
[                          <parameter>]
[                              <name>UseWildcard</name>]
[                              <value>true</value>]
[                          </parameter>]
[                      </search-option>]
[                  </simple-search-page>]
[              </search-pages>]
[              <search-result-pages>]
[                  <search-result-list-page>]
[                      <search-result-id>1</search-result-id>]
[                      <item-per-page>10</item-per-page>]
[                      <max-result-size>100</max-result-size>]
[                      <field-group>]
```

```
[                   <description/>]
[                   <field-ref>Person.FirstName</field-ref>]
[                   <field-ref>Person.MiddleName</field-ref>]
[                   <field-ref>Person.LastName</field-ref>]
[                   <field-ref>Person.SSN</field-ref>]
[                   <field-ref>Person.DOB</field-ref>]
[                   <field-ref>Person.Gender</field-ref>]
[                 </field-group>]
[             </search-result-list-page>]
[         </search-result-pages>]
[     </record-details>]
[     <reports>]
[         <root-object>Person</root-object>]
[         <tab-name>Reports</tab-name>]
[         <screen-id>6</screen-id>]
[         <display-order>5</display-order>]
[         <search-pages/>]
[         <search-result-pages/>]
[         <subscreen-configurations>]
[             <subscreen>]
[                 <enable>true</enable>]
[                 <root-object>Person</root-object>]
[                 <tab-name>Potential Duplicate Report</tab-name>]
[                 <report-name>Potential Duplicate</report-name>]
[                 <screen-id>0</screen-id>]
[                 <display-order>1</display-order>]
[                 <search-pages/>]
[                 <search-result-pages>]
[                     <search-result-list-page>]
[                         <search-result-id>0</search-result-id>]
[                         <item-per-page>10</item-per-page>]
[                         <max-result-size>2000</max-result-size>]
[                         <field-group>]
[                             <description/>]
[                             <field-ref>Person.FirstName</field-ref>]
[                             <field-ref>Person.LastName</field-ref>]
[                             <field-ref>Person.SSN</field-ref>]
[                             <field-ref>Person.DOB</field-ref>]
[                             <field-ref>Person.Gender</field-ref>]
[                         </field-group>]
[                     </search-result-list-page>]
[                 </search-result-pages>]
[             </subscreen>]
[             <subscreen>]
[                 <enable>true</enable>]
[                 <root-object>Person</root-object>]
[                 <tab-name>Activity Report</tab-name>]
[                 <report-name>Transaction Summary Report</report-name>]
[                 <screen-id>2</screen-id>]
[                 <display-order>2</display-order>]
[                 <search-pages>]
[                     <simple-search-page>]
[                         <screen-title>Weekly Activity</screen-title>]
[                         <report-name>Weekly Transaction Summary
Report</report-name>]
[                         <search-result-id>0</search-result-id>]
[                         <search-screen-order>1</search-screen-order>]
[                         <field-group/>]
[                     </simple-search-page>]
[                     <search-result-pages>]
```

```
[                              <search-result-list-page>]
[                                  <search-result-id>0</search-result-id>]
[                                  <item-per-page>10</item-per-page>]
[                                  <max-result-size>2000</max-result-size>]
[                                  <field-group/>]
[                              </search-result-list-page>]
[                          </search-result-pages>]
[                  </subscreen>]
[        </page-definition>]
[    </gui-definition>]
[]]]]]]
```

3

Master Person Index Field Notations

This chapter introduces you to Oracle Healthcare Master Person Index field notations and provides examples of each type of notation.

This chapter includes the following section:

- "Learning About Master Person Index Field Notations"

Learning About Master Person Index Field Notations

The configuration files use specific notations to define a specific field or a group of fields in an enterprise or system object. There are three different types of notations used by Oracle Healthcare Master Person Index.

The following sections describe each type of notation used:

- "ePath Notation"
- "Qualified Field Name Notation"
- "Simple Field Name Notation"

ePath Notation

In update.xml, an **element path**, called an ePath, is used to specify the location of a field or list of fields. ePaths are also used in the StandardizationConfig element of mefa.xml. An ePath is a sequence of nested nodes in an enterprise record where the most nested element is a data field or a list of data fields. ePaths allow you to retrieve and transform values that are located in the object tree.

ePath strings can be of four basic types:

- **ObjectField** - A field defined in the master person index object structure.
- **ObjectNode** - A parent or child object defined in the master person index object structure.
- **ObjectField List** - A list of references to certain ObjectFields in the master person index object structure.
- **ObjectNode List** - A list of references to certain ObjectNodes in the master person index object structure.

A context node is specified when evaluating each ePath expression. The context is considered as the root node of the structure for evaluation.

These topics describe and illustrate how to form ePath strings:

- "ePath Syntax"

■ "ePath Notation Example"

ePath Syntax

The syntax of an ePath consists of three components: nodes, qualifiers, and fields, as shown below.

```
node{.node{"["qualifier']'}+}+.field
```

- **Node** - Specifies the node type and optionally includes qualifiers to restrict the number of nodes. A node without any qualifier defaults to only the first node of the specified type. Use "node.*" to address a node rather than a field.

- **Qualifier** - Restricts the number of nodes addressed at each level. The following qualifiers are allowed:

  - ***** (asterisk) - Denotes all nodes of the specified type.

  - **int** - Accesses the node by index.

  - **@keystring= valuestring** - Accesses the node using a key-value pair. Only one instance of the node is addressed using keys. If a composite key is defined, then multiple key-value pairs can be separated by a comma in the ePath (for example, [@key1=value1,@key2=value2]). The following ePath uses the keystring qualifier and returns the alias where the unique key field type is "Main". It returns only one alias in a given record.

    Person.Alias[@type=Main]

  - **filter=value** - Considers only nodes whose field matches the specified value. A subset of nodes is addressed using filters. Multiple filter-value pairs can be separated by a comma (for example, [filter1=value1, filter2=value2]). The following ePath uses the filter qualifier and returns all aliases where the last name is "Jones".

    ```
    Person.Alias[lastname=Jones]
    ```

- **Field** - Designates the field to return and is in the form of a string.

ePath Notation Example

The following sample illustrates an object structure containing a system object from Site A with a local ID of 111. The object contains a first name, last name, and three addresses. Following the sample, there are several ePath examples that refer to various elements of this object structure along with a description of the data in the sample object structure referred by each ePath.

```
Enterprise
   SystemObject - A 111
      Person
         FirstName
         LastName
         -Address
            AddressType = Home
            Street = 800 Royal Oaks Dr.
            City = Monrovia
            State = CA
            PostalCode = 91016
         -Address
            AddressType = Office
            Street = 181 2nd Ave..
            City = Monrovia
            State = CA
```

```
                    PostalCode = 91016
                -Address
                    AddressType = Billing
                    Street = 100 Grand Avenue
                    City = El Segundo
                    State = CA
                    PostalCode = 90245
```

- Person.Address.City - Equivalent to Person.Address[0].City.

- Person.FirstName - Uses Person as the context, and is equivalent to Enterprise.SystemObject[@SystemCode=A, @Lid= 111].Person.FirstName with Enterprise as the context.

- Person.Address[@AddressType=Home].City - Returns a single ObjectField reference to "Monrovia" (the City field of the home address).

- Person.Address[City=Monrovia,State=CA].Street - Returns a list of ObjectField references: "800 Royal Oaks Dr.", "181 2nd Ave." (the street fields for both addresses where the city is Monrovia and the state is CA). Note that a reference to the Billing address is not returned.

- Person.Address[*].Street - Returns a list of ObjectField references: "800 Royal Oaks Dr.", "181 2nd Ave.", "100 Grand Avenue". Note that all references to Street are returned.

- Person.Address[2].* - Addresses the second address object as an ObjectNode instead of an ObjectField.

Qualified Field Name Notation

In query.xml and the MatchingConfig element of mefa.xml use qualified field names to specify the location of a field. This method defines a specific field and is not used to define a list of fields. A qualified field name is a sequence of nested nodes in an enterprise record where the most nested element is a data field.

There are two types of qualified field names.

- **Fully qualified field names** - Allow you to define fields within the context of the enterprise object; that is, the field name uses Enterprise as the root. These are used in the MatchingConfig element of mefa.xml and to specify the fields in a query block in query.xml.

- **Qualified field names** - Allow you to define fields within the context of the parent object; that is, the field name uses the name of the parent object as the root. These are used in query.xml to specify the source fields for the blocking query criteria.

The following topics describe and illustrate how to form qualified field name strings.

- "Qualified Field Name Syntax"

- "Qualified Field Name Example"

Qualified Field Name Syntax

The syntax of a fully qualified field name is:

```
Enterprise.SystemSBR.parent_object.child_object.field_name
```

where parent_object refers to the name of the parent object in the index, child_object refers to the name of the child object that contains the field, and field_name is the full name of the field. If the parent object contains the field being defined, the child object is not required in the path.

The syntax of a qualified field name is:

```
parent_object.child_object.field_name
```

Qualified Field Name Example

The following sample illustrates an object structure that could be defined in object.xml. The object contains a Person parent object, and Address and Phone child objects.

```
Person
    FirstName
    LastName
    DateOfBirth
    Gender
    -Address
        AddressType
        StreetAddress
        Street
        City
        State
        PostalCode
    -Phone
        PhoneType
        PhoneNumber
```

The following fully qualified field names are valid for the sample structure above.

- Enterprise.SystemSBR.Person.FirstName

- Enterprise.SystemSBR.Person.Address.StreetAddress

- Enterprise SystemSBR.Person.Phone.PhoneNumber

The qualified field names that correspond with the fully qualified names listed above are:

- Person.FirstName

- Person.Address.StreetAddress

- Person.Phone.PhoneNumber

Simple Field Name Notation

In midm.xml, simple field names are used to specify the location of a field that appears on the MIDM. These are used in the GUI configuration section of the file. Simple field names define a specific field and are not used to define a list of fields. They include only the field name and the name of the object that contains the field. Simple field names allow you to define fields within the context of an object.

The following topics describe and illustrate how to form simple field notations:

- "Simple Field Notation Syntax" on page 3-4

- "Simple Field Notation Example" on page 3-5

Simple Field Notation Syntax

The syntax of a simple field name is:

```
object.field_name
```

where object refers to the name of the object that contains the field being defined and field_name is the full name of the field.

Simple Field Notation Example

The following sample illustrates an object structure that could be defined in object.xml. The object contains a Person parent object, and Address and Phone child objects.

```
Person
    FirstName
    LastName
    DateOfBirth
    Gender
    -Address
        AddressType
        StreetAddress
        Street
        City
        State
        PostalCode
    -Phone
        PhoneType
        PhoneNumber
```

The following simple field names are valid for the sample structure above.

- Person.FirstName

- Address.StreetAddress

- Phone.PhoneNumber