Oracle® Healthcare Master Person Index

Configuration Guide

Release 3.0

E62294-01

March 2015

ORACLE®

Oracle Healthcare Master Person Index Configuration Guide, Release 3.0

E62294-01

Contents

3 Master Person Index Structure, Properties, and Restrictions

4 Working With Master Person Index Queries

5 Processing Options, Matching Parameters, and EUIDs

6 Normalization, Standardization, and Phonetic Encoding

7 Master Person Index Survivor Calculator

8 Processes, Update Policies, and Field Validations

9 Master Index Data Manager Configuration

Preface

The *Oracle Healthcare Master Person Index Configuration Guide* provides conceptual and procedural information for configuring the Oracle Healthcare Master Person Index and the Master Index Data Manager. Chapter 1, "Security Configuration Issues," describes security configuration issues you should consider when implementing OHMPI.

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.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

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

- *Oracle Healthcare Master Person Index Configuration Guide* [This document]
- *Oracle Healthcare Master Person Index Analyzing and Cleansing Data User's Guide*
- *Oracle Healthcare Master Person Index Australia Patient Solution User's Guide*
- *Oracle Healthcare Master Person Index Command Line Reports and Database Management User's Guide*
- *Oracle Healthcare Master Person Index Configuration Reference*
- *Oracle Healthcare Master Person Index Data Manager User's Guide*
- *Oracle Healthcare Master Person Index Installation Guide*

- *Oracle Healthcare Master Person Index Loading the Initial Data Set User's Guide*
- *Oracle Healthcare Master Person Index Match Engine Reference*
- *Oracle Healthcare Master Person Index Message Processing Reference*
- *Oracle Healthcare Master Person Index Provider Index User's Guide*
- *Oracle Healthcare Master Person Index Real-time Loader User's Guide*
- *Oracle Healthcare Master Person Index Release Notes*
- *Oracle Healthcare Master Person Index Security Guide*
- *Oracle Healthcare Master Person Index Standardization Engine Reference*
- *Oracle Healthcare Master Person Index United Kingdom Patient Solution User's Guide*
- *Oracle Healthcare Master Person Index United States Patient Solution User's Guide*
- *Oracle Healthcare Master Person Index User's Guide*
- *Oracle Healthcare Master Person Index Working With HPD Profile Application User's Guide*
- *Oracle Healthcare Master Person Index Working With IHE Profiles User's Guide*

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

Finding Information and Patches on My Oracle Support

Your source for the latest information about Oracle Healthcare Master Person Index is Oracle Support's self-service Web site My Oracle Support (formerly MetaLink).

Before you install and use Oracle Healthcare Master Person Index, always visit the My Oracle Support Web site for the latest information, including alerts, White Papers, installation verification (smoke) tests, bulletins, and patches.

Creating a My Oracle Support Account

You must register at My Oracle Support to obtain a user name and password account before you can enter the Web site.

To register for My Oracle Support:

1. Open a Web browser to `https://support.oracle.com`.
2. Click the **Register here** link to create a My Oracle Support account. The registration page opens.
3. Follow the instructions on the registration page.

Signing In to My Oracle Support

To sign in to My Oracle Support:

1. Open a Web browser to `https://support.oracle.com`.
2. Click **Sign In.**
3. Enter your user name and password.
4. Click **Go** to open the My Oracle Support home page.

Finding Information on My Oracle Support

There are many ways to find information on My Oracle Support.

Searching by Article ID

The fastest way to search for information, including alerts, White Papers, installation verification (smoke) tests, and bulletins is by the article ID number, if you know it.

To search by article ID:

1. Sign in to My Oracle Support at https://support.oracle.com.
2. Locate the Search box in the upper right corner of the My Oracle Support page.
3. Click the sources icon to the left of the search box, and then select **Article ID** from the list.
4. Enter the article ID number in the text box.
5. Click the magnifying glass icon to the right of the search box (or press the Enter key) to execute your search.

   The Knowledge page displays the results of your search. If the article is found, click the link to view the abstract, text, attachments, and related products.

Searching by Product and Topic

You can use the following My Oracle Support tools to browse and search the knowledge base:

- Product Focus — On the Knowledge page under Select Product, type part of the product name and the system immediately filters the product list by the letters you have typed. (You do not need to type "Oracle.") Select the product you want from the filtered list and then use other search or browse tools to find the information you need.
- Advanced Search — You can specify one or more search criteria, such as source, exact phrase, and related product, to find information. This option is available from the **Advanced** link on almost all pages.

Finding Patches on My Oracle Support

Be sure to check My Oracle Support for the latest patches, if any, for your product. You can search for patches by patch ID or number, or by product or family.

To locate and download a patch:

1. Sign in to My Oracle Support at https://support.oracle.com.
2. Click the **Patches & Updates** tab. The Patches & Updates page opens and displays the Patch Search region. You have the following options:
   - In the **Patch ID or Number is** field, enter the number of the patch you want. (This number is the same as the primary bug number fixed by the patch.) This option is useful if you already know the patch number.
   - To find a patch by product name, release, and platform, click the **Product or Family** link to enter one or more search criteria.
3. Click **Search** to execute your query. The Patch Search Results page opens.
4. Click the patch ID number. The system displays details about the patch. In addition, you can view the Read Me file before downloading the patch.

5. Click **Download.** Follow the instructions on the screen to download, save, and install the patch files.

Finding Oracle Documentation

The Oracle Web site contains links to all Oracle user and reference documentation. You can view or download a single document or an entire product library.

Finding Oracle Health Sciences Documentation

To get user documentation for Oracle Health Sciences applications, go to the Oracle Health Sciences documentation page at:

http://www.oracle.com/technetwork/documentation/hsgbu-154445.html

---

Note: Always check the Oracle Health Sciences Documentation page to ensure you have the latest updates to the documentation.

Finding Other Oracle Documentation

To get user documentation for other Oracle products:

1. Go to the following Web page:

   http://www.oracle.com/technology/documentation/index.html

   Alternatively, you can go to http://www.oracle.com, point to the Support tab, and then click **Documentation**.

2. Scroll to the product you need and click the link.
3. Click the link for the documentation you need.

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

Security Configuration Issues

This chapter describes security configuration issues you should consider when implementing OHMPI:

General Security Principles

The following principles are fundamental to using any application securely.

Keep Software Up To Date

One of the principles of good security practice is to keep all software versions and patches up to date.

Keep Up To Date on Latest Security Information Critical Patch Updates

Oracle continually improves its software and documentation. Critical Patch Updates are the primary means of releasing security fixes for Oracle products to customers with valid support contracts. We highly recommend customers apply these patches as soon as they are released.

Configure Strong Passwords on the Database

Although the importance of passwords is well known, the following basic rule of security management is worth repeating:

Ensure all passwords are strong passwords.

You can strengthen passwords by creating and using password policies for your organization. For guidelines on securing passwords and for additional ways to protect passwords, refer to the *Oracle® Database Security Guide* specific to the database release you are using.

You should modify the following passwords to use your policy-compliant strings:

- Passwords for the database default accounts, such as SYS and SYSTEM.
- Passwords for the database application-specific schema accounts, such as RXI.
- The password for the database listener. Oracle recommends that you do not configure a password for the database listener as that will enable remote administration. For more information, refer to the section "Removing the Listener Password" of *Oracle® Database Net Services Reference 12c Release 1 (12.1)*.

Follow the Principle of Least Privilege

The principle of least privilege states that users should be given the least amount of privilege to perform their jobs. Overly ambitious granting of responsibilities, roles, grants — especially early on in an organization's life cycle when people are few and work needs to be done quickly — often leaves a system wide open for abuse. User privileges should be reviewed periodically to determine relevance to current job responsibilities.

Managing Default User Accounts

Lock and expire default user accounts.

Closing All Open Ports Not in Use

Keep only the minimum number of ports open. You should close all ports not in use.

Disabling the Telnet Service

Oracle Healthcare Master Person Index standard configuration does not use the Telnet service.

Telnet listens on port 23 by default.

If the Telnet service is available on any computer, Oracle recommends that you disable Telnet in favor of Secure Shell (SSH). Telnet, which sends clear-text passwords and user names through a log-in, is a security risk to your servers. Disabling Telnet tightens and protects your system security.

Disabling Other Unused Services

In addition to not using Telnet, the Oracle Healthcare Master Person Index standard configuration does not use the following services or information for any functionality:

- Simple Mail Transfer Protocol (SMTP). This protocol is an Internet standard for E-mail transmission across Internet Protocol (IP) networks.
- Identification Protocol (identd). This protocol is generally used to identify the owner of a TCP connection on UNIX.
- Simple Network Management Protocol (SNMP). This protocol is a method for managing and reporting information about different systems.

Restricting these services or information does not affect the use of Oracle Healthcare Master Person Index standard configuration. If you are not using these services for other applications, Oracle recommends that you disable these services to minimize your security exposure. If you need SMTP, identd, or SNMP for other applications, be sure to upgrade to the latest version of the protocol to provide the most up-to-date security for your system.

Designing for Multiple Layers of Protection

When designing a secure deployment, design multiple layers of protection. If a hacker should gain access to one layer, such as the application server, that should not automatically give them easy access to other layers, such as the database server.

Providing multiple layers of protection may include:

- Enable only those ports required for communication between different tiers, for example, only allowing communication to the database tier on the port used for SQL*NET communications (1521 by default).
- Place firewalls between servers so that only expected traffic can move between servers.

Utilizing SSL

Consider utilizing Application Server SSL service for the MPI application. The MPI application is a standard J2EE application and it can utilize an industry standard security infrastructure and framework. There is no configuration needed on the MPI application itself. The WebLogic application server provides SSL service. Refer to the application server's documentation to configure SSL to achieve SSL security for MPI.

Other OHMPI Security Issues that You Should Consider

- Master Index Data Manager (MIDM) roles and permissions

  For information about roles and permissions, see the *Oracle Healthcare Master Person Index Data Manager User's Guide*.

Removing OHMPI Installer PL/SQL Script

For security reasons, Oracle recommends that you delete the following PL/SQL installation scripts that are no longer needed:

- create.sql
- codelist.sql
- systems.sql
- create-index.sql
- drop-index.sql

These scripts are located at
NetbeansProjectlocation/ProjectName/src/DatabaseScript/

2

Configuration Files Overview

This chapter introduces you to the Oracle Healthcare Master Person Index (OHMPI) configuration files, and informs you which XML files you can edit directly or with the OHMPI Configuration Editor. It also presents conceptual information for maintaining version control of the configuration files and provides rules you should follow when copying, cutting, and pasting files.

This chapter includes the following sections:

Introducing the Master Person Index Configuration Files

Oracle Healthcare Master Person Index provides a very flexible framework for creating a master person index application that is customized for your requirements. An Oracle Healthcare Master Person Index project includes several files in XML format that define the configuration of the runtime environment. You can configure a master person index application by modifying the XML files directly or by using the Configuration Editor. Make sure to verify the configuration of the application before deploying the project.

> **Note:** It is helpful to review the information provided in the *Oracle Healthcare Master Person Index Configuration Reference* to learn about the relationships between the files and what components and processes can be configured. Certain components can only be configured by modifying the XML files directly

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.

object.xml

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.

query.xml

In `query.xml`, you configure the **Query Builder** component of the master person index application and define the available queries. In this file, you define the types of queries that can be performed from the Master Index Data Manager (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.

mefa.xml

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. This file 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 phonetically encoded. 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 standardization and match engine in use. *Oracle Healthcare Master Person Index Match Engine Reference* and *Oracle Healthcare Master Index Standardization Engine Reference* describe the rules for the OHMPI Match Engine and OHMPI Standardization Engine.

You can also configure portions of the match process in `master.xml`, described below, which defines certain match parameters that control weight thresholds, how assumed matches are processed, how potential duplicates are processed, and the query to use for matching.

master.xml

In `master.xml`, you configure the **Manager Service** and define 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.

This file 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.

update.xml

In `update.xml`, you 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 entity.

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.

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`.

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.

validation.xml

By default, `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 master person index application.

security.xml

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

midm-security.xml

For information on the midm-security.xml file, see the *Oracle Healthcare Master Person Index Data Manager User's Guide.*

midm.xml

In `midm.xml`, you configure the appearance and processing properties of the Master Person Index Data Manager (MIDM). 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.

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 MIDM. These files configure the runtime components of the master person index application.

match-ext.xml

The `match-ext.xml` file provides the capability to create variations in sets of match fields and agreement/disagreement weights at runtime. Using frequency-based agreement weights adjustment, field value swapping, weight cap on a set of selected fields, and then based upon the type of data you provide, it enables the OHMPI Match Engine to compute more accurate matches when a specific behavior is desired.

This file extends the matching functionality you can perform on your OHMPI projects.

Note: For information about the new matching enhancements in the OHMPI Match Engine, see the *Oracle Healthcare Master Person Index Match Engine Reference*.

Modifying the Master Person Index XML Files Directly

Make sure that when you modify the configuration files, you use the versioning controls in NetBeans to maintain version control. Even if you modify the files using the OHMPI Configuration Editor, check the files in and out before and after using the editor.

There are a few restraints on modifying these files. In addition to the general rules listed below, the match or standardization engine you choose might place other requirements on customizations. Be sure to review *Oracle Healthcare Master Person Index Match Engine Reference* and *Oracle Healthcare Master Index Standardization Engine Reference* before modifying `mefa.xml`.

Keep the following guidelines in mind when modifying the XML files directly.

- All fields specified in any of the configuration files must be included in `object.xml`.
- If you add fields to the object structure, make sure you add them to the survivor calculator in `update.xml`.
- If you define additional fields for normalization, parsing, or phonetic encoding, make sure to add the normalized, parsed, and phonetic fields to `object.xml` and, optionally, the blocking query.
- After modifying any of the configuration files, you must regenerate the master person index application and redeploy the project. You must also refresh any client projects that reference the master person index server project.

Using the Master Person Index Configuration Editor

The Configuration Editor has built in validations to ensure that integrity is maintained between the configuration files. For example, it does not allow you to define a field for normalization if that field is not already defined in the object structure. While you can use the Configuration Editor to modify most of the configurable components, some components can only be modified using the XML editor. Following is a summary of which features can be configured using the Configuration Editor and which need to be modified using an XML editor.

object.xml

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 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; that needs to be done manually.

query.xml

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

master.xml

Most elements in `master.xml` cannot be modified using the Configuration Editor. You can only 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 (either through the Configuration Editor or XML editor) 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.

validation.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.

midm.xml

Most 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 (matchConfigFile.cfg)

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.

Maintaining Version Control in the Master Person Index Configuration Files

Whether you modify the XML files directly or configure the master index application through the OHMPI Configuration Editor, be sure to maintain version control by checking files out before modifying them and checking them back in after you are finished. You can use any of the NetBeans supported platforms for version control. When working with the Configuration Editor, make sure to check out each file that will be modified, either directly or indirectly. For example, if you add a new field to the object structure, add the field to the match string, and add the field to a blocking query, the Configuration Editor updates `object.xml`, `query.xml`, and `mefa.xml`.

Copying, Cutting, and Pasting Files

You can use standard cut, copy, and paste commands to copy or move files between projects. OHMPI follows the standard NetBeans functionality, with the exception that you can only copy or move a component from one project into the same node of another project. For example, you can only paste a copied configuration file into the Configuration node of another project. In addition, you cannot cut components that are essential to a project, such as the configuration files, match and standardization files, and so on.

3

Master Person Index Structure, Properties, and Restrictions

This chapter provides conceptual information and procedures for configuring Oracle Healthcare Master Person Index (OHMPI) object structures. It also provides information on field properties and field name restrictions.

This chapter includes the following sections:

Configuring the Master Person Index Object Structure

After you create the master person index framework and create the configuration files, you can modify the object structure that you defined. The properties for the objects and fields you will store in the master person index database are defined in `object.xml`. For more information about this file and the configurable options, see "Master Person Index Object Definition Configuration" in *Oracle Healthcare Master Person Index Configuration Reference*.

The object tree in the OHMPI Configuration Editor corresponds to `object.xml` in the master person index application. Any changes you make to the object structure in the Configuration Editor are reflected in `object.xml`, and MIDM field property changes are also reflected in `midm.xml`. If you modify the XML file directly, you need to update `midm.xml` manually.

Changing the object structure might also require that you make corresponding changes to the other configuration files. For example, if you add a new field to `object.xml` that you want to include in queries and matching, you need to make the corresponding changes to `query.xml`, `mefa.xml`, and `update.xml`. Only changes made before generating the project take effect for the new application.

You can configure the object structure either through the OHMPI Configuration Editor or by modifying the XML file directly. Both methods are described here. Perform any of the following actions to configure the object structure.

Adding an Object to the Master Person Index Object Structure

You can add new objects to the object structure as needed. Note that the object structure can contain only one parent object but multiple child objects. If you use the Configuration Editor, you can either create an undefined object or create an object from a predefined template. A predefined object includes a set of predefined fields with default configurations.

Note: Due to database naming constraints, the length of the name of the parent object plus the length of any child object names must be 21 characters or less.

To Add an Undefined Object (Configuration Editor)

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Select the parent object, and then click **Add Sub Object Node** on the Configuration Editor toolbar.

   The object structure expands, and a new child object appears at the bottom of the object structure.

3. Type a new name for the object, and then press **ENTER**.

4. Define the fields for the new object, as described in Adding a Field to the Master Person Index Object Structure on page 3-5.

5. On the Configuration Editor toolbar, click **Save**.

To Add a Predefined Object (Configuration Editor)

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Select the parent object, click **Templates** in the toolbar, and then select the template you want to use.

   The new child object and any defined fields appear in the object tree.

3. To change the name of the new object, double-click the object name, type the new name, and then press **Enter**.

4. Do any of the following:

   - Add fields to the new object, as described in Adding a Field to the Master Person Index Object Structure on page 3-5.
   - Delete fields from the new objects, as described in Deleting a Field from the Master Person Index Object Structure on page 3-6.

- Modify the properties of the fields in the object, as described in Modifying Master Field Properties on page 3-7.

5. On the Configuration Editor toolbar, click **Save**.

To Add an Undefined Object (XML editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **object.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the location where you want to create the new object (after the database element but before relationships).

3. Create a nodes element.

4. Create and name a tag element within the new nodes element (the value of the tag element is the name of the object you are defining).

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

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

5. Define the fields for the new object, as described in Adding a Field to the Master Person Index Object Structure on page 3-5.

6. Define the relationship of the new object to the existing objects, as described in Defining Relationships Between Master Person Index Objects on page 3-8.

7. Save and close the file.

Modifying an Object's Name In the Master Person Index Object Definition

You can modify the name of an object in `object.xml`, but you must also make the corresponding changes to the remaining configuration files.

To Modify an Object's Name (Configuration Editor)

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **plus sign** next to the parent object to expand the object structure.

3. Click twice on the name of the object you want to change.

4. Type in the new name, and then press **Enter**.

5. On the Configuration Editor toolbar, click **Save**.

To Modify an Object's Name (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **object.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the tag element defining the object you want to modify.
3. Change the value of the tag element.
4. Modify the object name in the relationships definition, as described in Defining Relationships Between Master Person Index Objects on page 3-8.
5. Save and close the file.

Deleting an Object From the Master Person Index Object Structure

If you define an object in error, you can remove the object from `object.xml`. If you modify the XML file directly, you must also remove the relationship definition for the object. Remember to make the corresponding changes to the remaining configuration files.

To Delete an Object (Configuration Editor)

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.
2. Expand the object structure by clicking the plus sign by the parent object.
3. Do any of the following:
   - Right-click in the **object tree panel**, and then click **Delete**.
   - Press the **Delete key**.
   - In the Configuration Editor toolbar, click **Delete**.
4. On the confirmation dialog box, click **Yes**.

   The object and any fields associated with that object are deleted. If you remove the parent object, all child objects are also removed.
5. On the Configuration Editor toolbar, click **Save**.

To Delete an Object (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **object.xml**.

   The file opens in the NetBeans XML editor.
2. Scroll to the nodes element containing the object you want to delete.
3. Delete all text between and including the nodes tags that contain the object tag.

   For example, to delete the Address object below, delete the boldface text.

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

4. Remove the object name from the relationship list, as described in Defining Relationships Between Master Person Index Objects on page 3-8.
5. Save and close the file.

Adding a Field to the Master Person Index Object Structure

Once you define an object in `object.xml`, you can add new fields to the object and configure the properties for those fields. Be sure to add the field to any relevant structures in the remaining configuration files. For information about field naming restrictions, see Master Person Index Field Name Restrictions on page 3-10.

To Add a Field (Configuration Editor)

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Expand the object structure until you see the object to which you want to add a field.

3. In the object tree panel, do one of the following:

   - To add the field to the end of the object's field list, select the name of the object to which you want to add a new field and then click **Add Field** in the toolbar.
   - To add the field immediately following an existing field, select the field after which you want to add the new field and then click **Add Field** in the toolbar.

     The tree expands and a new field is inserted.

4. To change the field name, double-click the new field, type the new name, and then press **Enter**.

5. Continue to Modifying Master Field Properties on page 3-7.

To Add a Field (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **object.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the tag element defining the object to which you want to add a field.

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

   For example:

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

4. Specify the field properties described in Master Person Index Field Property Elements on page 3-12 within the new fields tags.

   For example:

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

```
</fields>
```

5. Save and close the file.

Deleting a Field from the Master Person Index Object Structure

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

WARNING: It is advisable to delete a field in the Configuration Editor as it makes all the corresponding changes to the files for you. If you delete a fields in the XML Editor, you will need to make the corresponding changes manually. Not making all the corresponding changes in the XML Editor can cause problems.

To Delete a Field (Configuration Editor)

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Expand the object structure until the field you want to delete is visible.
3. Select the field and perform one of the following:
   - Right-click in the **object tree panel**, and then click **Delete**.
   - Press the **Delete key.**
   - In the Configuration Editor toolbar, click **Delete**.
4. On the confirmation dialog, click **Yes**.

   The field is removed from the object tree.

5. On the Configuration Editor toolbar, click **Save**.

To Delete a Field (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **object.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the tag element defining the object from which you want to delete a field.
3. Scroll to the *fields* element containing the field to delete, and then delete all text between and including the fields tags defining that field.

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

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

```
</fields>
```

4. Save and close the file.

Modifying Master Field Properties

Every field in the object structure has a set of properties that must be configured before deploying the master person index application. When a field is created, a set of default properties are defined for that field. You can modify the property configuration for each field to suit your data processing, storage, and display requirements. Field properties include general field attributes, such as the name, length, and data type, and MIDM field properties, such as the display name for the field and display formatting.

> **Note:** On the Configuration Editor, the fields in the MIDM section cannot be configured if the field is not defined to appear on the MIDM (that is, it does not appear in `midm.xml`).

On the Configuration Editor, the fields in the MIDM section cannot be configured if the field is not defined to appear on the MIDM (that is, it does not appear in `midm.xml`).

To Modify Field Properties (Configuration Editor)

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Expand the object structure until the field you want to configure is visible.

3. In the object structure, select the field to configure.

   The General Properties page appears.

4. On the Properties page in the right side of the window, modify the value of any of the properties listed in Master Person Index Configuration Editor Field Properties on page 3-10.

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

5. To view or modify matching properties, click the **Matching** tab and configure the properties as described in To Configure the Comparison Functions (Configuration Editor) on page 5-12.

   This tab is only visible for fields that have a value in the Match Type field.

6. On the Configuration Editor toolbar, click **Save.**

To Modify Field Properties (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **object.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the tag element defining the object to modify, and then to the *fields* element containing the field to modify.
3. Modify the value of any of the elements described in Master Person Index Field Property Elements on page 3-12.

> **Note:** If you modify the name of a field, make the corresponding changes to the remaining configuration files. Some property elements might not exist for a field; add any necessary elements to the field definition to configure the field.

4. Save and close the file.

Example 3–1 Field Properties in object.xml

The following example defines an address type field that is required in order to enter a record, and that uniquely identifies each address object in a record. It also defines a list, named ADDRTYPE, from which MIDM users can select a value to enter into the field.

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

The following example defines an employee ID field where the <minimum value> must be equal to or greater than 100001 and less than or equal to 199999. Only the characters 0-9 are allowed.

```
<field-name>EmployeeID</field-name>
<field-type>string</field-type>
<size>6</size>
<updateable>true</updateable>
<required>true</required>
<minimum-value>100001</minimum-value>
<maximum-value>199999</maximum-value>
<pattern>[0-9]{6}</pattern>
<key-type>false</key-type>
```

Defining Relationships Between Master Person Index Objects

Once all objects are customized, you must define relationships between those objects if you are modifying the XML file directly. If you are using the Configuration Editor, the relationships are automatically defined in the object tree.

To Define Object Relationships (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **object.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the relationships element near the end of the file.

3. To specify a new parent object, modify the value of the name element.

   For example:

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

   Note: This is not recommended. Changing the parent name requires changing all instances of the name in all configuration files. To change the parent object name, use the Configuration Editor, which automatically propagates the change.

4. To change the name of a child object, modify the value of the appropriate children element.
5. To add a child object, create and name a new children element.
6. To delete a child object, delete all text between and including the appropriate children element.
7. Save and close the file.

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

Understanding Object Name Restrictions

Do not use the following object names because it will cause an error in the create script (due to a clash between the generated object name and a standard OHMPI table) or in the Java code (because the generated method in the ObjectNode extension will clash with an existing method):

- SYSTEM
- SYSTEMS
- SYSTEMOBJECT
- ENTERPRISE
- TRANSACTION
- ASSUMEDMATCH
- OVERWRITE
- POTENTIALDUPLICATES
- AUDIT
- MERGE
- APPL
- COMMON_HEADER
- COMMON_DETAIL
- SEQ_TABLE
- Child

Defining Master Person Index Field Properties and Understanding Name Restrictions

Once you create the object structure of the master person index application, you can customize several properties for each field, such as whether the field is required, will include a drop-down menu on the MIDM, will be used in a blocking query, and so on. There are also some restrictions for how fields can be named and how the properties are defined on the Configuration Editor and in the configuration files.

The following sections provide information about the naming restrictions and about the field properties of the wizard.

Master Person Index Field Name Restrictions

When you name the fields in your object structure, be sure to keep the following guidelines in mind to avoid errors when compiling or running the master person index application.

- Oracle Healthcare Master Person Index automatically creates a field for each object named objectId, where object is the name of an object or sub-object. You cannot create fields with those names. For example, you cannot create a field named "AddressId" if there is an Address object in the object structure.
- If you enter a field name longer than 20 characters, a warning dialog box appears. While most databases can handle names up to at least 30 characters, Oracle Healthcare Master Person Index appends text to the end of fields defined for phonetic encoding or standardization. For fields that will be parsed, normalized, or phonetically encoded, make sure the name of the original field does not exceed 20 characters. Any other field can have a name up to 30 characters long. For information about the names of the fields automatically created by the wizard, see the *Oracle Healthcare Master Person Index Message Processing Reference*.
- In field names, do not use characters or names restricted by Java, XML, or the database platform you are using.

Master Person Index Configuration Editor Field Properties

The following table lists and describes the properties you can define on the Field Properties page of the Configuration Editor.

Property	Description
Name	A name for the field. See Master Person Index Field Name Restrictions on page 3-10 for information on valid field names.
Data Type	The data type for each field. Possible data types are: ■ **string** - Field of this type contain a string of characters. ■ **date** - Fields of this type contain a date value. ■ **float** - Fields of this type contain a floating point integer. ■ **int** - Fields of this type contain an integer. ■ **char** - Fields of this type contain a single character. ■ **boolean** - Fields of this type can contain either **true** or **false**.

Property	Description
Match Type	The type of matching to be performed against the field, if the field is to be used for match weight generation. You must define at least one field for matching or no weights will be generated. **Note**: The match types you specify here define the structure of `mefa.xml`, including the match string. The match types in `mefa.xml` might differ from the match types specified here. See the *Oracle Healthcare Master Person Index Message Processing Reference* for information about the available options for this field.
Blocking	An indicator of whether the field will be used in the blocking query. Select this option to add the field to the blocking query; deselect this option to omit the field from the blocking query.
Key Type	An indicator of whether the field is used to identify unique objects. For example, a business index might store several addresses for each business. Each address is assigned an address type and each business can only have one address of each type. Select this option if the field is a unique record identifier; otherwise, deselect it. Key type fields should also be required fields unless combinations of fields are specified as key types for an object. **Note**: Oracle recommends that each child object must contain a key type field. If child objects do not contain one or more key type fields, each enterprise object might accumulate a very large number of child objects depending on the survivor strategy used.
Updateable	An indicator of whether the field can be updated from the MIDM and external system messages. Select this option if the field can be updated; otherwise deselect it.
Required	An indicator of whether the field is required in order to save an enterprise object to the database. Select this option if the field is required; otherwise, deselect it.
Field Size	The number of characters allowed in each field. This determines the number of characters allowed in the database columns and defines the maximum number of characters that can be entered into each field on the MIDM.
Pattern	The required data pattern for the field. For more information about possible values and using Java patterns, see "Patterns" in the class list for java.util.regex in the Javadocs provided with the Java EE Platform. You might want to define patterns for date, telephone, or SSN fields. Note that for the MIDM, the pattern is further restricted by the value entered for the input mask described later in this table. If no input mask is specified, all regex patterns are supported.
Code Module	The identification code for the drop-down list that appears for this field in the MIDM. **Note**: This must match an entry in the **code** column of the sbyn_common_ header database table and, by default, an entry for the code you enter is created in the **Code List** database script. You can further customize code lists directly in the Code List script.
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.
Constrained By	The name of the field that contains the corresponding User Code value (described above) used to validate the current field. The User Code and Constrained 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 Constrained By value against definitions for the field with the user code value. For example, if you store non-unique IDs such as credit card numbers or insurance policy numbers, you could create a field named **ID Type** 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**.

Property	Description
Display Name	The name of the field as it will appear on the MIDM.
Input Mask	A mask used by the GUI to add punctuation to a field. For example, if users enter the date in the format MMDDYYYY, you can add an input mask to display the dates as MM/DD/YYYY. Use the value mask (described below) to strip the punctuation from the value before storing it in the database. To define an input mask, type a character type for each character in the field and place any necessary punctuation between the character types. For example, the input mask for the above date format is DD/DD/DDDD. **Note**: The value you enter can further restrict the data pattern for the field (this is the Pattern property). The following character types can be used. **D** - indicates a numeric character. **L** - indicates an alphabetic character. **A** - indicates an alphanumeric character.
Value Mask	A mask used by the index to strip any extra characters that were added by the input mask (see above). This mask ensures that data is stored in the database in the correct format. To specify a value mask, type the same value as is entered for the input mask, but type an "x" in place of each punctuation mark. For example, if an SSN field has an input mask of DDD-DD-DDDD, specify a value mask of DDDxDDxDDDD to strip the dashes before storing the SSN. A value mask is not required for date fields.
Print Confidential	You can print the SBR from the Record Details page. By default, it is set to true. ■ If print-confidential is set to true, the print report shows a Confidential tag at the top of the report. ■ If print-confidential is set to false, then following are the two possibilities: ■ If there is any field that is set to sensitive, then irrespective of print-confidential set to false, the Confidential tag is added to the top of the print report. ■ If there are no fields that are set to sensitive, then the Confidential tag is not added to the top of the report.
Show Result Set	By default, it is set to false. ■ If show-resultset is set to false, the potential duplicate report displays the records in the default vertical format. ■ If show-resultset is set to true, the potential duplicate report displays the records in the tabular format.
Is Summary Display	If the is-summary-display for a field is set to true, then that field is added to the summary display shown on the EUID Details screen.

Master Person Index Field Property Elements

The following table lists and describes the properties you can define for each field in `object.xml`.

Element	Description
field-name	The name of the field. Follow the guidelines under Master Person Index Field Name Restrictions on page 3-10 when naming fields.

Element	Description
field-type	The data type of the field. Possible values 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. **char** - 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 element, be sure to modify the length of the corresponding database column accordingly.
undateable	An indicator of whether the field can be updated from the MIDM or by 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_comment_header database table. This element is optional.
maximum-value	The maximum value allowed in the field. To specify a value for a date field, use the format YYYY-MM-DD. This element is optional.
minimum-value	The minimum value allowed in the field. To specify a value for a date field, use the format YYYY-MM-DD. This element is optional.
pattern	The required pattern for the field. For more information about possible values and using Java patterns, see "Patterns" in the class list for java.util.regex in the Javadocs provided with J2SE Platform. You might want to define patterns for date, telephone, or SSN fields. Note that for the MIDM, the pattern is further restricted by the value entered for the input mask. If no input mask is specified, all regex patterns are supported. 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) used 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**.
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. Key type fields should also be required fields unless a combination of fields are specified as key types for an object. This element is optional. **Note**: Each child object should contain at least one field that is a unique object identifier, but this 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.

4

Working With Master Person Index Queries

This chapter provides instructions on how to create and modify basic and blocking queries. It also provides procedures for deleting queries that are no longer required.

This chapter includes the following sections:

Creating a Master Person Index Basic Query

By default, two basic queries are predefined in query.xml, one phonetic and one exact alphanumeric. If the default queries do not meet your query requirements, you can define new queries for the master person index application. You can either use an existing query builder class or create a custom query builder by creating a custom plug-in. For more information, see the *Oracle Healthcare Master Person Index User's Guide*.

The changes you make on the Query page of the Configuration Editor are reflected in query.xml. For more information about this file and the configurable query options, see the *Oracle Healthcare Master Person Index Configuration Reference*. If you create a new query to use from the MIDM, make sure to add that query to one of the search definitions in midm.xml. Unless specifically defined for range searching in midm.xml, basic queries use exact searching. No configuration is required in query.xml for basic exact or range searching.

You can create basic queries either through the Configuration Editor or by modifying the XML file directly. Both methods are described here.

To Create a Basic Query (Configuration Editor)

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **Query** tab.

   The Query page appears.

3. In the Basic Queries section, click **Add**.

   The Basic Query Builder dialog box appears.

4. Enter values and select options for the fields described in Master Person Index Query Builder Dialog Box Fields and XML Elements on page 4-3.
5. On the Configuration Editor toolbar, click **Save**.

To Create a Basic Query (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **query.xml**.

   The file opens in the NetBeans XML editor.

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

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

3. For the new query-builder element, define the attributes listed in Master Person Index Query Builder Dialog Box Fields and XML Elements on page 4-3.

   For example:

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

4. To configure the query to use wildcard characters, do the following:
   - Add a new config element after the opening query-builder element.

     For example:

```
<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="true">
   <config>
   </config>
</query-builder>
```

5. In the new config element, create an option element and then define key and value attributes for the new element.

   For example:

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

> **Note:** For the default basic query, only the UseWildcard parameter is supported. If you create a custom basic query builder, you can use these elements to define additional parameters. For more information, see Master Person Index Query Builder Dialog Box Fields and XML Elements on page 4-3.

6. Save and close the file.

Master Person Index Query Builder Dialog Box Fields and XML Elements

The following table lists and describes the fields on the Basic Query Builder and Blocking Query Builder dialog boxes on the Configuration Editor, along with the corresponding elements and attributes in query.xml.

Configuration Editor Field	XML Element/Attribute	Description
Query Builder Name	query-builder/name	A unique ID that identifies 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 the name.
Query Builder Class	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 to not 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.
Parser Class	query-builder/ parser-class	The fully qualified name of the class that parses the configuration elements for each query. The default parser class is com.sun.mdm.index.configurator.impl.querybuilder. KeyValueConfiguration.
Standardize	query-builder/ standardize	An indicator of whether any of the query criteria is standardized before being passed to the query. On the Configuration Editor, select this check box if any search fields are standardized. In the XML file, enter **true** if any search fields are standardized; otherwise enter **false**.
Phoneticize	query-builder/ phoneticize	An indicator of whether any of the query criteria is phonetically encoded before being passed to the query. On the Configuration Editor, select this check box if any search fields are phonetically encoded. In the XML file, enter **true** if any search fields are phonetically encoded; otherwise enter **false**.
Use Wildcard	option/key option/value	An indicator of whether wildcard characters are allowed when executing this search. This parameter is available for basic queries only. On the Configuration Editor, select this check box if wildcard characters are allowed in any of the search fields. In the XML file, each option element configures one parameter for the query and contains a key and value pair. The key attribute names the parameter. For the default basic query, the name is UseWildCard. The value attribute specifies the value of the corresponding key attribute. For the UseWildCard parameter, specify **true** to allow wildcard characters for that query type; otherwise specify **false**. When wildcard characters are enabled, you can enter a percent sign (%) into search criteria to represent multiple unknown characters. **Note**: If you define a custom query, you can use key and value options to define any required parameters.

Creating Master Person Index Blocking Queries

By default, one blocking query is predefined in query.xml. If the default query does not meet your requirements, you can define new queries for the master person index application. You can either use an existing query builder class or create a custom query builder by creating a custom plug-in (for more information, see "Implementing Master Person Index Custom Plug-ins" in *Oracle Healthcare Master Person Index User's Guide*). Blocking queries contain **block definitions**, which define the groups of fields used for each query pass and how those groups are processed. Each block definition contains one or more set of **query blocks**. Each query block defines the query rules for one of the groups of fields that make up the block definition. You can further configure a blocking query by filtering out unwanted values from the query process. For more information, see Filtering Default Values From Master Person Index Processes on page 8-1.

The changes you make on the Query page of the Configuration Editor are reflected in query.xml. For more information about this file and the configurable query options, see "Configuring Queries" in *Oracle Healthcare Master Person Index Configuration Reference*. If you create a new query to use from the MIDM, make sure to add that query to one of the search definitions in midm.xml.

You can create blocking queries either through the Configuration Editor or by modifying the XML file directly. Both methods are described here.

To Create a Blocking Query (Configuration Editor)

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **Query** tab.

   The Query page appears.

3. In the Blocking Queries section, click **Add**.

   The Blocking Query Builder dialog box appears.

4. In the Query Builder section, enter the fields described in Master Person Index Query Builder Dialog Box Fields and XML Elements on page 4-3.

5. Do the following for each query block you want to include in the query:

   a. In the Block Definitions section, click **Add**.

      The Block Definition dialog box appears.

   b. In the **Block Name** field, enter a unique name for the query block.

   c. (Optional) In the **Hint** field, define Oracle hints for the query block.

   d. In the Block By section, click **Add.**

      The Block Rule dialog box appears, where you can specify a field to include in the query block.

   e. Enter values for the fields on the dialog box as described in Master Person Index Query Builder Dialog Box Fields and XML Elements on page 4-3, and then click **OK**.

   f. For each field to include in the query block, repeat the previous two steps.

g. When you are done specifying fields for the rule, click **OK** on the Block Definition dialog box.

6. For each query block you want to create for this query, repeat the previous step.
7. When you are done defining the query blocks, click **OK**.
8. On the Configuration Editor toolbar, click **Save**.
9. To define values to exclude as criteria during the query process, see Filtering Default Values From Master Person Index Processes on page 8-1.

To Create a Blocking Query (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **query.xml**.

   The file opens in the NetBeans XML editor.

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

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

3. For the new query-builder element, define the attributes listed in Master Person Index Query Builder Dialog Box Fields and XML Elements on page 4-3.

   For example:

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

4. Create a new config element between the query-builder tags.
5. To create a query block, do the following:

   - In the new config element, create a new block-definition element with a number attribute and assign a unique ID code to the number attribute.

     For example:

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

   - To add a database hint, create and define a new hint element in the new block-definition element.

     The following example illustrates an Oracle hint.

```
<config>
   <block-definition number="ID1">
   <hint>FIRST_ROWS_100</hint>
```

```
   </block-definition>
<config>
```

Note: Hints are especially useful when a blocking query uses only child object fields; the hint can specify to scan the child object table first.

- In the new block-definition element, create a block-rule element.
- For each field in the data block, create the elements and attributes listed in Master Person Index Query Block Fields and XML Elements on page 4-6.

  The following example illustrates the use of both range and equals elements, as well as upper and lower bounds.

```
<config>
   <block-definition number="ID1">
      <hint>FIRST_ROWS_100</hint>
      <block-rule>
         <equals>
            <field>Enterprise.SystemSBR.Person.FirstNamePh</field>
            <source>Person.FirstNamePh</source>
         </equals>
         <equals>
            <field>Enterprise.SystemSBR.Person.LastNamePh</field>
            <source>Person.LastNamePh</source>
         </equals>
         <range>
            <field>Enterprise.SystemSBR.Person.DateOfBirth</field>
            <source>Person.DateOfBirth</source>
            <default>
               <lower-bound type="offset">-5</lower-bound>
               <upper-bound type="constant">2009-01-01
               </upper-bound>
            </default>
         </range>
      </block-rule>
   </block-definition>
<config>
```

6. Repeat the previous step for each data block you need to define for the query.

   All data blocks for one query must be defined within one config element.

7. Save and close the file.
8. To define values to exclude as criteria during the query process, see Filtering Default Values From Master Person Index Processes on page 8-1.

Master Person Index Query Block Fields and XML Elements

The following table lists and describes the fields on the Block Rule dialog box of the Configuration Editor, along with the corresponding elements and attributes in query.xml. To see how the XML elements are organized, see the sample in Adding a Query Block to a Master Person Index Query on page 4-10. For more information about the structure of a query block, see the *Oracle Healthcare Master Person Index Configuration Reference*.

Configuration Editor Field	XML Element/Attribute	Description
Field	field	The name of the field to be included in the query block. On the Configuration Editor, you can click Browse to select a field or enter the fully qualified field name manually.

Source	source	The name of the source field in the object from which the criteria is obtained. Click Browse to select a field, or enter the qualified field name manually. An asterisk (*) can be used as a wildcard character.
		Tip: When a field in a child object is defined for a blocking query, use the asterisk wildcard character in the ePath to the source field to ensure all instances of the child object in an incoming message are used as search criteria. Each instance is joined by an OR operator. For example, this configuration:
		`<field>Enterprise.SystemSBR.Person.Name.FirstName`
		`</field>`
		`<source>Person.Name[*].FirstName</source>`
		would result in a WHERE clause similar to this:
		WHERE Name.FirstName="Meg" OR Name.FirstName="Maggie"

Operator	type of search	An indicator of the type of search to perform on the field. On the Configuration Editor, you can select one of the following values from the drop-down list. In the XML file, you specify one of the following names for the XML element that defines one field in a query block. ■ **equals** - Performs an exact search against either the criteria or the value defined for the "Use Constants" option. ■ **not-equals** - Searches for values that do not equal either the criteria or the value defined for the "Use Constants" option. ■ **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 "Use Constants" option. ■ **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 "Use Constants" option. ■ **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. 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.

Configuration Editor Field	XML Element/Attribute	Description
Use Constant/Value	constant	On the Configuration Editor, this is an indicator of whether to use a constant value as the search criteria for the field. When this option is selected, you need to enter the constant in the corresponding Value field. In the XML file, enter the value of the constant in the constant element.
Upper Bound Type	upper-bound/type	For range searching only, the type of value to use for the upper limit of the search range. Select or enter one of the following options. ■ **not defined** - No specific upper limit is defined; a user enters the value when performing the search. Be sure to define this field for range searching in the MIDM as well. In the XML file, this is indicated by omitting the upper-bound element. ■ **constant** - A specific value is defined to use as the upper limit of the range when no search criteria is entered or when incomplete information is available. ■ **offset** - A value to be added to the user-supplied value to determine the upper limit on the search range. For constant and offset, enter the value in the corresponding Value field.
Lower Bound Type	lower-bound/type	For range searching only, the type of value to use for the lower limit of the search range. Select one of the options listed for Upper Bound Type. If you select offset, the value you specify for the offset will be subtracted from the user-supplied value.
Value (for the upper and lower bounds)	upper-bound lower-bound	For constant and offset range searching, the upper or lower limit of the range. For constants, this is the upper or lower limit; for offsets, this is the value added to or subtracted from the user-supplied value. It can be numeric, date, or string. In the XML file, the upper-bound and lower-bound elements fall within an element named default.

Modifying Master Person Index Queries

By default, two basic queries and one blocking query are predefined in query.xml and you can create additional queries. Once a query is defined, you can modify it as needed. If you make changes to a query defined for a search in the MIDM, you might need to modify the search definition in midm.xml. For example, if you deselect the Use Wildcard option for a query and the wildcard feature is enabled for that query in the MIDM, you need to disable that feature in midm.xml.

You can customize the queries used in your implementation by performing any of the following actions.

- Modifying a Master Person Index Query on page 4-9
- Adding a Query Block to a Master Person Index Query on page 4-10
- Modifying a Query Block for a Master Person Index Query on page 4-12
- Deleting a Query Block From a Master Person Index Query on page 4-13

You can also define exclusion lists to filter out unwanted values from a blocking query. For more information, see Filtering Default Values From Master Person Index Processes on page 8-1.

Modifying a Master Person Index Query

Once you define a basic query, you can modify the attributes and parameters for that query. If you defined a custom query, you might need to modify certain elements of the query directly in the XML file instead of through the Configuration Editor.

To Modify a Basic Query (Configuration Editor)

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **Query** tab.

   The Query page appears.

3. Do one of the following:

   - To modify a blocking query, select the query to modify in the Blocking Queries section, and then click **Edit**.

     The Blocking Query Builder dialog box appears.

   - To modify a basic query, select the query to modify in the Basic Queries section, and then click **Edit**.

     The Basic Query Builder dialog box appears.

4. Modify any of the fields and configuration options described in Master Person Index Query Builder Dialog Box Fields and XML Elements on page 4-3.

5. For blocking queries, you can modify query blocks by performing any of the following procedures:

   - Adding a Query Block to a Master Person Index Query on page 4-10
   - Modifying a Query Block for a Master Person Index Query on page 4-10
   - Deleting a Query Block From a Master Person Index Query on page 4-13

6. When you are done editing the query, click **OK**.

7. On the Configuration Editor toolbar, click **Save**.

8. To modify or define values to be excluded as criteria (for blocking queries only), go to Filtering Default Values From Master Person Index Processes on page 8-1.

To Modify a Query (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **query.xml**.

   The file opens in the NetBeans XML editor.

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

3. Modify any of the attributes listed in Master Person Index Query Builder Dialog Box Fields and XML Elements on page 4-3.

   For example:

   ```
   <query-builder name="ALPHA-SEARCH" class=
   ```

```
  "com.sun.mdm.index.querybuilder.MyQueryBuilder"
  parser-class=
  "com.sun.mdm.index.configurator.impl.querybuilder.KeyValueConfiguration"
   standardize="true" phoneticize="false">
</query-builder>
```

4. To add a new query parameter (for custom query builders only):
   - Add a new config element between the query-builder tags.
   - In the new config element, create an option element and then define key and value attributes for the new element.

     For example:

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

     See Master Person Index Query Builder Dialog Box Fields and XML Elements on page 4-3 for information about the key and value pairs available to the default basic queries.
5. To modify a parameter, scroll to the option element that defines the parameter and then change the value of the value attribute.
6. To delete a parameter, scroll to the config element of that query, and then delete the option element containing the parameter to remove.

   For example, to remove the SearchAlias parameter in the following sample, delete the **boldface text**.

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

7. For blocking queries, you can modify query blocks by performing any of the following procedures:
   - Adding a Query Block to a Master Person Index Query on page 4-10
   - Modifying a Query Block for a Master Person Index Query on page 4-12
   - Deleting a Query Block From a Master Person Index Query on page 4-13
8. Save and close the file.
9. To modify or define values to be excluded as criteria (for blocking queries only), go to Filtering Default Values From Master Person Index Processes on page 8-1.

Adding a Query Block to a Master Person Index Query

Some query blocks might be predefined for you based on information you specified in the wizard. You can create additional query blocks for a blocking query.

To Add a Query Block (Configuration Editor)

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.
2. Click the **Query** tab.

The Query page appears.

3. In the Blocking Queries section, select the query you want to modify and then click **Edit**.

   The Blocking Query Builder dialog box appears.

4. In the Block Definitions section, click **Add**.

   The Block Definition dialog box appears.

5. In the **Block Name** field, enter a unique name for the query block.

6. (Optional) In the **Hint** field, define a database hint for the query block.

7. In the Block By section, click **Add**.

   The Block Rule dialog box appears, where you can specify a field to include in the query block.

8. Enter values for the fields on the dialog box as described in Master Person Index Query Block Fields and XML Elements on page 4-6, and then click **OK**.

9. Repeat the previous two steps for each field to add to the query block.

10. When you are done, click **OK** on the Block Definition dialog box.

11. On the Blocking Query Builder dialog box, click **OK**.

12. On the Configuration Editor toolbar, click **Save**.

To Add a Query Block (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **query.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the query-builder element that contains the blocking query to configure.

3. If a config element does not exist for the query, create one between the query-builder tags.

4. In the config element, create a new block-definition element with a number attribute and assign a unique ID code to the number attribute.

   For example:

```
<config>
      <block-definition number="ID1">
      </block-definition>
   </config>
</query-builder>
```

5. To add a hint, create and define a new hint element in the new block-definition element.

   The following example illustrates an Oracle hint.

```
<config>
   <block-definition number="ID1">
   <hint>FIRST_ROWS_100</hint>
   </block-definition>
<config>
```

6. In the new block-definition element, create a block-rule element.

7. For each field in the data block, define the elements described in Master Person Index Query Block Fields and XML Elements on page 4-6.

   For example:

```
<config>
   <block-definition number="ID1">
      <hint>FIRST_ROWS_100</hint>
      <block-rule>
         <equals>
            <field>Enterprise.SystemSBR.Person.FirstNamePh</field>
            <source>Person.FirstNamePh</source>
         </equals>
         <equals>
            <field>Enterprise.SystemSBR.Person.LastNamePh</field>
            <source>Person.LastNamePh</source>
         </equals>
         <range>
            <field>Enterprise.SystemSBR.Person.DateOfBirth</field>
            <source>Person.DateOfBirth</source>
            <default>
               <lower-bound type="offset">-5</lower-bound>
               <upper-bound type="constant">2006-01-01
               </upper-bound>
            </default>
         </range>
      </block-rule>
   </block-definition>
<config>
```

8. Save and close the file.

Modifying a Query Block for a Master Person Index Query

Once block definitions are created for a blocking query, you can modify those definitions if needed. You can add, modify, and remove block fields. When adding or removing fields from a query block, verify that the query will still include all the needed fields.

To Modify a Query Block (Configuration Editor)

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **Query** tab.

   The Query page appears.

3. In the Blocking Queries section, select the query you want to modify and then click **Edit**.

   The Blocking Query Builder dialog box appears.

4. In the Block Definitions section, select the block to modify and then click **Edit**.

   The Block Definition dialog box appears.

5. Do any of the following:

   - To add a new field, click **Add**. Enter the fields described in Master Person Index Query Block Fields and XML Elements on page 4-6, and then click **OK**.

- To edit an existing field, select the field you want to modify, click Edit, modify any of the fields described in Master Person Index Query Block Fields and XML Elements on page 4-6, and then click **OK**.
- To delete an existing field, select the field you want to delete, click **Remove**, and then click **Yes** on the dialog box that appears.

6. When you are done modifying the query block, click **OK**.
7. On the Blocking Query Builder dialog box, click **OK**.
8. On the Configuration Editor toolbar, click **Save**.

To Modify a Query Block (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **query.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the block-rule element containing the query block you want to modify.
3. To add a new blocking field, add and populate the elements described in Master Person Index Query Block Fields and XML Elements on page 4-6.

   For example:

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

4. To edit an existing field, modify any of the elements described in Master Person Index Query Block Fields and XML Elements on page 4-6.
5. To delete an existing field, delete all text between and including the search type elements that define the field (such as equals, range, not-equals, and so on).

   For example, to delete the DOB field defined above, you would delete the text between and including the equals elements.

6. Save and close the file.

Deleting a Query Block From a Master Person Index Query

Once you create a block definition, you can delete it if needed. When deleting a query block, verify that the query will still include all the needed fields.

To Delete a Query Block (Configuration Editor)

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **Query** tab.

   The Query page appears.

3. In the Blocking Queries section, select the query you want to modify and then click **Edit**.

   The Blocking Query Builder dialog box appears.

4. In the Block Definitions section, select the definition you want to delete.
5. Click **Remove**.
6. On the confirmation dialog box, click **Yes**.
7. On the Configuration Editor toolbar, click **Save**.

To Delete a Query Block (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **query.xml**.

   The file opens in the NetBeans XML editor.
2. Scroll to the query-builder element that contains the blocking query to modify.
3. In that query-builder element, delete all text between and including the block-definition element defining the data block to remove.

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

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

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

4. Save and close the file.

Deleting a Master Person Index Query

Once you create a query, you can delete it if needed. If any of the queries you delete are defined as the matching query in master.xml or as a search in midm.xml, modify those files to point to an existing query. If a defined query is no longer used, you can leave the query in query.xml as it will not affect processing if you do not delete the query.

> **Note:** The changes you make on the Query page of the Configuration Editor are reflected in query.xml. For more information about this file and the configurable query options, see the *Oracle Healthcare Master Person Index Configuration Reference*.

The changes you make on the Query page of the Configuration Editor are reflected in query.xml. For more information about this file and the configurable query options, see the *Oracle Healthcare Master Person Index Configuration Reference*.

To Delete a Query (Configuration Editor)

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **Query** tab.

   The Query page appears.

3. In the Blocking Queries or Basic Queries section, select the query you want to delete.

4. Click **Remove**.

5. On the Configuration Editor toolbar, click **Save**.

To Delete a Query

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **query.xml**.

   The file opens in the NetBeans XML editor.

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

3. Delete all text between and including the query-builder element defining the query, including any defined query blocks.

   For example, to delete the query named PHONETIC-SEARCH, you would remove all text below.

```
<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>
```

4. Save and close the file.

5

Processing Options, Matching Parameters, and EUIDs

This chapter provides information and procedures for configuring processing options, matching parameters (including the Match Engine, and EUIDs).

This chapter includes the following sections:

Configuring Master Person Index Processing Options

The Master Controller defines how transactions are processed, and is configured in master.xml. You can specify Java classes that insert additional logic into the match process, configure the blocking query to use for matching, define how merged record updates are handled, and specify whether a record's potential duplicates are reevaluated after it is updated. You can customize the Master Controller by performing any of the following actions.

Specifying Master Person Index Custom Logic Classes

The logic-class element specifies custom match processing logic for messages coming from external systems. The logic-class and logic-class-gui elements specify custom match processing logic for the MIDM. If no custom plug-ins were created to define the custom logic, leave these elements empty. Custom logic classes can only be specified by modifying the XML file directly.

To Specify Custom Logic for External System Messages

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **master.xml**.
2. In the MasterControllerConfig element, change the value of the logic-class element to the name of the custom plug-in that contains the back-end logic.

   For example:

   ```
   <logic-class>com.sun.mdm.index.user.CustomProcessing</logic-class>
   ```

3. Save and close the file.

To Specify Custom Logic for the Master Person Index Data Manager

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **master.xml**.
2. In the MasterControllerConfig element, change the value of the logic-class-gui element to the name of the custom plug-in that contains the MIDM logic.

   For example:

   ```
   <logic-class-gui>com.sun.mdm.index.user.CustomMIDM</logic-class-gui>
   ```

3. Save and close the file.

Specifying the Master Person Index Update Mode

The update-mode element specifies whether potential duplicates are reevaluated for a record each time the record is updated. If you specify that potential duplicates are reevaluated, the reevaluation only occurs when updates are made to fields involved in blocking and matching. The update mode can only be specified by modifying the XML file directly.

To Specify Potential Duplicates be Reevaluated at Each Update

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **master.xml**.
2. In the MasterControllerConfig element, change the value of the update-mode element to `Pessimistic`.

   For example:

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

3. Save and close the file.

To Specify Potential Duplicates not be Reevaluated at Each Update

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **master.xml**.
2. In the MasterControllerConfig element, change the value of the update-mode element to `Optimistic`.

   For example:

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

3. Save and close the file.

Configuring Master Person Index Merged Record Updates

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

To Allow Merged Record Updates

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **master.xml**.
2. In the MasterControllerConfig element, change the value of the merged-record-update element to `Enabled`.

   For example:

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

3. Save and close the file.

To Prevent Merged Record Updates

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **master.xml**.
2. In the MasterControllerConfig element, change the value of the merged-record-update element to `Disabled`.

   For example:

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

3. Save and close the file.

Specifying the Master Person Index Blocking Query for Matching

You need to specify the query that will be used by the master person index application to retrieve a candidate selection pool of records that closely match an incoming record and that will be used for probabilistic weighting. The name of the blocking query you specify here must match the name of one of the blocking queries defined in query.xml. You can only specify one blocking query for matching in the master person index application.

> **Note:** Modifying the blocking query once your system is in production is not recommended because it can cause issues with data integrity.

To Specify the Blocking Query for Matching

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the execute-match element in the MasterControllerConfig element.
3. Modify the value of the query-builder name to the name of the blocking query you want to use. For example:

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

Note: Make sure the name you specify exactly matches the name of one of the blocking queries defined in query.xml.

4. Set the blocking query options, as described in Setting Master Person Index Blocking Query Options on page 5-4.
5. Save and close the file.

Setting Master Person Index Blocking Query Options

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

To Set Blocking Query Options

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the execute-match element.
3. To specify a key and value pair, modify the values of the key and value attributes in the option element.

   For example:

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

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

Defining Master Person Index Transactional Support

You can define 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. You can define transactional support by using the Configuration Editor or by modifying the XML file directly.

To Configure MPI for Local Transactions

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.
2. Click the **Deployment tab**.
3. In the **Transaction Mode** field, select **LOCAL**.
4. On the Configuration Editor toolbar, click **Save**.
5. To configure the database connection, launch the application service Admin Console, and do one of the following:
   - If the database connection is already defined, do the following on the Admin Console:

  - In the left frame, expand **Resources > JDBC > Connection Pools**.
  - Select the database connection pool that is defined for your master person index application.
  - On the Edit Connection Pool page, select either **javax.sql.DataSource** or **javax.sql.ConnectionPoolDataSource** for the Resource Type and click **Save**.
- If the database connection is not already defined, see the *Oracle Healthcare Master Person Index User's Guide*. Select either **javax.sql.DataSource** or **javax.sql.ConnectionPoolDataSource** for the connection pool Resource Type.

To Configure MPI to Distributed Transactions Across Applications

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit.**

   The Configuration Editor appears.
2. Click the **Deployment tab**.
3. In the **Transaction Mode** field, select **XA (across applications).**
4. On the Configuration Editor toolbar, click **Save**.
5. Launch the application server Admin Console (from NetBeans, click the **Services tab**, expand **Servers**, right-click the **domain**, and select **View Admin Console**).
6. To configure the database connection, launch the **application service Admin Console**, and do one of the following:
   - If the database connection is already defined, do the following on the Admin Console:
     - In the left frame, expand **Resources > JDBC > Connection Pools**.
     - Select the database connection pool that is defined for your master person index application.
     - On the Edit Connection Pool page, select **javax.sql.XADataSource** for the Resource Type and click **Save**.
   - If the database connection is not already defined, follow the instructions provided in *Oracle Healthcare Master Person Index User's Guide*. Select **javax.sql.XADataSource** for the Resource Type.

To Configure MPI for Distributed Transactions Within the MPI Application Only

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.
2. Click the **Deployment tab**.
3. In the **Transaction Mode** field, select **XA (within application only)**.
4. On the Configuration Editor toolbar, click **Save.**
5. To configure the database connection, launch the **application server Admin Console** and do one of the following:
   - If the database connection is already defined, do the following on the Admin Console:
     - In the left frame, expand **Resources > JDBC > Connection Pools**.

  - Select the database connection pool that is defined for your master person index application.
  - On the Edit Connection Pool page, select **javax.sql.XADataSource** for the Resource Type and click **Save**.
- If the database connection is not already defined, follow the instructions provided in the *Oracle Healthcare Master Person Index User's Guide*. Select javax.sql.XADataSource for the Resource Type.

Configuring Matching Parameters

The Decision Maker is configured in master.xml and defines how to handle certain decision points in the matching process, such as weight thresholds, how to handle multiple records above the match threshold, and whether records originating from the same system can be automatically matched.

You can customize the Decision Maker by performing any of the following actions:

Specifying the Master Person Index Decision Maker Class

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

To Specify the Decision Maker Class

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the **DecisionMakerConfig element.**
3. Change the value of the decision-maker-class element to the path and name of the new Decision Maker class.

   For example:

   ```
   <decision-maker-class>com.sun.mdm.index.decision.impl.MyDecisionMaker
   </decision-maker-class>
   ```

4. Save and close the file.

Defining How to Handle Multiple Assumed Matches (OneExactMatch)

The OneExactMatch parameter determines how records above the match threshold are processed. This parameter can only be modified directly in the XML file. For more information, see *Oracle Healthcare Master Person Index Configuration Reference* and Oracle Healthcare Master Person Index *Message Processing Reference*.

To Create Potential Duplicates When Multiple Records Match

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the **OneExactMatch element** in the DecisionMakerConfig element.
3. Change the value of the parameter-value element to **true**.

   For example:

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

4. Save and close the file.

To Match the Highest Weighted Records When Multiple Records Match

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the **OneExactMatch element** in the DecisionMakerConfig element.
3. Change value of the parameter-value element to **false**.

   For example:

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

4. Save and close the file.

Specifying Whether Same System Matches are Allowed (SameSystemMatch)

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

To Allow Same System Records to be Automatically Merged

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the **SameSystemMatch element** in the DecisionMakerConfig element.
3. Change the value of the parameter-value element to **false**.

   For example:

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

4. Save and close the file.

To Prevent Same System Records From Being Automatically Merged

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the **SameSystemMatch element** in the DecisionMakerConfig element.
3. Change the value of the parameter-value element to **true**.

   For example:

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

4. Save and close the file.

Specifying the Master Person Index Duplicate Threshold

The duplicate threshold is the lowest matching probability weight at which two records are considered potential duplicates of one another. Any records with lower probability weights are not considered to be possible matches. Any records between the duplicate and match thresholds are flagged as potential duplicates, and must be resolved manually. You can configure the duplicate threshold by using the Configuration Editor or by modifying the XML file directly.

To Specify the Duplicate Threshold (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit.**

   The Configuration Editor appears.
2. Click the **Matching** tab.
3. In the **Duplicate Threshold** field, enter the lowest weight at which two records must be considered as a potential match.

   Click **Refresh Values** to automatically compute the duplicate threshold. This calculates the duplicate threshold using the latest available matching configurations updates. To enter the threshold value, you can either type it in the text field or select it from the list of authorized range of the threshold values available for a specific configuration.

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

   ---

4. On the Configuration Editor toolbar, click **Save**.

To Specify the Duplicate Threshold (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the **DuplicateThreshold element** in the DecisionMakerConfig element.
3. Change the value of the parameter-value element. For example:

```
<parameter>
```

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

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

4. Save and close the file.

Specifying the Master Person Index Match Threshold

The match threshold specifies the matching probability weight at which two records will be automatically merged, depending on the value of the OneExactMatch parameter and the number of records at or above the match threshold. You can configure the match threshold by using the Configuration Editor or by modifying the XML file directly.

To Specify the Match Threshold (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **Matching** tab.

3. In the **Match Threshold** field, enter the lowest weight at which two records must be considered a match.

   Click **Refresh Values** to automatically compute the match threshold. This calculates the threshold using the latest available matching configurations updates. To enter the match threshold value, you can either type it in the text field or select it from the list of authorized range of the threshold values available for a specific configuration.

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

4. On the Configuration Editor toolbar, click **Save**.

To Specify the Match Threshold (XML Editor)

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **master.xml**.

2. Scroll to the **MatchThreshold element** in the DecisionMakerConfig element.

3. Change the value of the parameter-value element.

   For example:

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

```
   <parameter-value>28.5</parameter-value>
</parameter>
```

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

4. Save and close the file.

Adding and Deleting Master Person Index Decision Maker Parameters

New parameters cannot be used with the default Decision Maker class and existing parameters should not be deleted. If you create a custom Decision Maker class, you might need to add new parameters or delete existing ones for the new class. You can only perform these tasks by modifying the XML file directly.

To Add a New Decision Maker Parameter

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the **DecisionMakerConfig element.**
3. Create a parameter element inside the parameters element, but outside any existing parameter elements. Define the following elements:
   - **parameter-name** - The name of the parameter.
   - **parameter-type** - The type of parameter. Valid values are java.lang.Long, java.lang.Short, java.lang.Byte, java.lang.String, java.lang.Integer, java.lang.Boolean, java.lang.Double, or java.lang.Float.
   - **parameter-value** - The value of the parameter.

     For example:

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

4. Save and close the file.

To Delete a Decision Maker Parameter

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the **DecisionMakerConfig** element.
3. To delete one parameter, remove all text between and including the parameter element.

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

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

4. To remove all parameters, remove all text between and including the starting and ending parameters element.
5. Save and close the file.

Configuring the Match Engine

You can configure the match engine by specifying the match engine to use and configuring the predefined comparison functions. You can also plug in custom standardization and matching rules. You only need to specify the match engine to use if you are using an engine other than the OHMPI Match Engine.

Perform any of these steps to configure the match engine:

- Specifying a Match Engine for the Master Person Index on page 5-11
- Configuring the Comparison Functions for a Master Person Index Application on page 5-12
- Importing Custom Comparison Functions on page 5-15
- Deleting a Custom Comparison Function on page 5-16

Specifying a Match Engine for the Master Person Index

Oracle Healthcare Master Person Index can support different match engines depending on the adapter configured to communicate with the engine. Default classes are provided for using the OHMPI Match Engine. You can implement a custom match engine along with custom adapters. The match engine configuration is defined by the matcher-api and matcher-config elements.

> **Note:** The default adapters for the OHMPI Match Engine are com.sun.mdm.index.matching.adapter.SbmeMatcherAdapter and com.sun.mdm.index.matching.adapter.SbmeMatcherAdapterConfig.

To Configure the Match Engine

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.
2. Scroll to the **matcher-api element** in the MatchingConfig section.

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

```
<matcher-api>
   <class-name>com.sun.mdm.index.matching.adapter.SbmeMatcherAdapter
   </class-name>
</matcher-api>
```

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

```
<matcher-config>
   <class-name>
    com.sun.mdm.index.matching.adapter.SbmeMatcherAdapterConfig
   </class-name>
</matcher-config>
```

5. Save and close the file.

Configuring the Comparison Functions for a Master Person Index Application

The match configuration file in the Match Engine node of the master person index project lists and defines the configuration for each match type based on the predefined comparison function for the OHMPI Match Engine. These match types can be applied to each field in the match string. You can modify the configuration of the existing match types, add new match types, and specify whether the match engine should use agreement and disagreement weights or m-probabilities and u-probabilities.

For more information about the structure of the match configuration file and the comparison functions you can use, see the *Oracle Healthcare Master Person Index Match Engine Reference*.

To Configure the Comparison Functions (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit.**

   The Configuration Editor appears.

2. Click the **Matching** tab.

   The Matching page appears with a list of fields defined for matching and a list of comparators that you can modify.

3. In the **Probability Type** field, select one of the following:

   - **Use Agree/Disagreement Weight Ranges** - Uses agreement and disagreement weights for matching. If agreement and disagreement weights are used, the **m-probability** and **u-probability** fields are ignored and do not appear on the Matching page.
   - **Use M-Probabilities/U-Probabilities** - Uses m-probabilities and u-probabilities for matching. If m-probabilities and u-probabilities are used, the **agreement** and **disagreement weight** fields are ignored and do not appear on the page.

4. To add a new matching rule:

   - Click **Add** in the lower right portion of the window.

     The Edit Matching Rules dialog box appears.

- Fill in the fields described in Match Comparator Configuration Properties for Oracle Healthcare Master Person Index on page 5-13.
- Click **OK**.

5. To edit an existing matching rule:
   - Click **Edit** in the lower right portion of the window.

     The Edit Matching Rules dialog box appears.
   - Change the value of any of the fields described in Match Comparator Configuration Properties for Oracle Healthcare Master Person Index on page 5-13.
   - Click **OK**.
6. To remove an existing matching rule:
   - In the matching rule table, select the rule you want to delete.
   - Click **Remove**.
7. On the Configuration Editor toolbar, click **Save**.

To Configure the Comparison Functions (Text Editor)

1. In the project window, expand the **master person index project**, and then expand the **master person index application**.
2. In the Match Engine folder, double-click **matchConfigFile.cfg**.
3. In the **Probability Type** field, enter one of the following values:
   - **0** - Uses m-probabilities and u-probabilities for matching. If m-probabilities and u-probabilities are used, the **agreement** and **disagreement weight** fields are ignored.
   - **1** - Uses agreement and disagreement weights for matching. If agreement and disagreement weights are used, the **m-probability** and **u-probability** fields are ignored.
4. For each comparison function you want to configure, modify the value of any of the columns described in Match Comparator Configuration Properties for Oracle Healthcare Master Person Index on page 5-13.
5. Save and close the file.

Match Comparator Configuration Properties for Oracle Healthcare Master Person Index

The following table lists and describes the Configuration Editor fields used to define the comparison functions. It also lists the corresponding column in the match configuration file if you want to modify the file directly.

Configuration Editor Field	Match Configuration File Element and Column Number	Description
Match Type	match-type (column 1)	A value that indicates to the OHMPI Match Engine how each field should be weighted. Each field included in the match string (the MatchingConfig section of mefa.xml) must have a match type corresponding to a match type defined in this file.

Configuration Editor Field	Match Configuration File Element and Column Number	Description
Match Size	size (column 2)	The number of characters on which matching is performed, beginning with the first character. For example, to match on only the first four characters in a 10-digit field, the value of this column should be 4.
Null Field	null-field (column 3)	An index that specifies how to calculate the total weight for null fields or fields that only contain spaces. You can specify any of the following values. The Configuration Editor value is given first, followed by the match configuration file value in parenthesis. ■ **Zero weight (0)** - If one or both fields are empty, the weight used for the field is 0 (zero). ■ **Full Combination weight (1)** - If both fields are empty, the agreement weight is used; if only one field is empty, the disagreement weight is used. ■ **Full Agreement weight (a1)** - Specifies to use the full agreement weight if both fields are null. ■ **1/x of the Agreement weight (ax)** - Specifies to use the a fraction of the agreement weight if both fields are empty. The agreement weight is multiplied by the fraction 1/x to obtain the match weight for that field. When modifying the match configuration file directly, the default is **2** if no number is specified. You can specify any number from **1** through **10**. ■ **Full disagreement weight (d1)** - Specifies to use the full disagreement weight if both fields are null. ■ **1/x of the disagreement weight (dx)** - Specifies to use the disagreement weight if only one field is empty. The disagreement weight is multiplied by the fraction 1/x to obtain the match weight for the field. When modifying the match configuration file directly, the default is **2** if no number is specified. You can specify any number from **1** through **10**. In the above descriptions, the agreement and disagreement weights are either specified in this file or calculated using a logarithmic formula based on the m and u-probabilities (depending on the probability type).
Function	function (column 4)	The type of comparison to perform when weighting the field. For information about the available comparison functions, see the *Oracle Healthcare Master Person Index Match Engine Reference*.

Configuration Editor Field	Match Configuration File Element and Column Number	Description
Agreement Weight	agreement-weight (column 7)	The matching weight to be assigned to a field given that the fields match between two records; that is, the maximum match weight for a field. This number can be between **0** and **100** and can have up to **16 decimal points**. Only set this value if the Probability Type is set to use agreement and disagreement weights.
Disagreement Weight	disagreement-weight (column 8)	The matching weight to be assigned to a field given that the fields do not match between two records; that is, the minimum match weight for a field. This number can be between **0** and **-100** and can have up to **16 decimal points**. Only set this value if the Probability Type is set to use agreement and disagreement weights.
M-Probability	m-prob (column 5)	The initial probability that the specified field in two records will match if the records match. The probability is a double value between **0** and **1**, and can have up to **16 decimal points**. Only set this value if the Probability Type is set to use probabilities.
U-Probability	u-prob (column 6)	The initial probability that the specified field in two records will match if the records do not match. The probability is a double value between **0** and **1**, and can have up to **16 decimal points**. Only set this value if the Probability Type is set to use probabilities.
Extra Parameters	parameters (column 9)	Parameters correspond to the comparison function specified in the Function field or column. Some comparison functions do not take any parameters and some take multiple parameters. For information about which functions take parameters and the parameters they take, see the *Oracle Healthcare Master Person Index Match Engine Reference*.

Importing Custom Comparison Functions

The Master Index Match Engine is based on a very flexible framework that allows you to define new algorithms in the form of comparison functions that compare field values between two records. You need to import the comparison function into NetBeans to make it available to all master person index applications or only the current application.

This section only describes importing custom comparison functions after they have been created. For information about creating a custom comparison function, see the *Oracle Healthcare Master Person Index Match Engine Reference*.

To Import a Comparison Function

1. In the Projects window, expand the **main master person index project**.
2. Right-click Match Engine, and select **Import Comparator Plug-in**.
3. In the dialog box that appears, navigate to the location of the **plug-in ZIP file**.
4. Select the file containing the **plug-in**, and then click **Open**.

5. Do one of the following:
   - To import the plug-in and make it available to all future master person index application, click **Yes**.
   - To import the plug-in and make it only available to the current master person index application, click **No**.

   The contents of the ZIP file are imported into the Match Engine node and the new comparators are added to the list of comparator definitions in comparatorsList.xml.
6. In the Match Engine node, navigate to the **/lib folder** that was added and verify that all of the required files are there.
7. Open **comparatorsList.xml** and verify the new comparator definitions are included.

Deleting a Custom Comparison Function

If you add a custom comparison function to a master person index application in error, you can remove it from the Match Engine node. Optionally, you can simply make the comparison function unavailable by removing it from the comparators list in comparatorsList.xml.

To Delete a Custom Comparison Function

1. Back up the source files for the function in case you need them at a later time.
2. In the Project window, navigate to the **Match Engine node** in the master person index project and then to the **/lib folder.**
3. Right-click the folder containing the files to remove, and then select **Delete**.

   A confirmation dialog appears.
4. Click **Yes**.

   The source files are removed from the project.
5. Open **comparatorsList.xml** and remove the comparator definitions from the list.
6. Save and close the file.

Configuring Master Person Index EUIDs

The EUID Generator controls how the enterprise-wide unique identifiers (EUIDs) are created by the master person index application, including the length of the ID and the checksum value. The EUID generator can only be configured by modifying master.xml directly.

You can customize the EUID Generator by performing any of the following actions:

- Specifying the Master Person Index EUID Generator Class on page 5-17
- Specifying the Master Person Index EUID Length on page 5-17
- Specifying a Master Person Index Checksum Length on page 5-17
- Specifying the Master Person Index Chunk Size on page 5-18
- Adding and Deleting Master Person Index EUID Generator Parameters on page 5-18

Specifying the Master Person Index EUID Generator Class

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

To Specify the EUID Generator Class

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the **EuidGeneratorConfig element**.
3. Change the value of the euid-generator-class element to the path and name of the new EUID generator class.

   For example:

   ```
   <euid-generator-class>com.sun.mdm.index.idgen.impl.MyEuidGenerator
   </euid-generator-class>
   ```

4. Save and close the file.

Specifying the Master Person Index EUID Length

By default, the length of the EUIDs generated by the master person index application is 10 characters. You can modify this value if needed. You can also specify a new starting EUID number. For more information, see the *Oracle Healthcare Master Person Index User's Guide*.

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

To Specify the EUID Length

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the first parameter element, **IdLength**, in the EuidGeneratorConfig element.
3. Change the value of the parameter-value element to the desired length of the EUID.

   For example:

   ```
   <parameter>
      <parameter-name>IdLength</parameter-name>
      <parameter-type>java.lang.Integer</parameter-type>
      <parameter-value>12</parameter-value>
   </parameter>
   ```

4. Save and close the file.

Specifying a Master Person Index Checksum Length

A checksum value is a number appended to the end of each EUID that allows systems to validate EUIDs to ensure accurate identification of records throughout the network. For detailed information about checksum values and how they work, see the *Oracle Healthcare Master Person Index Configuration Reference*.

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

To Specify a Checksum Length

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the second parameter element, **ChecksumLength**, in the EuidGeneratorConfig element.
3. Change the value of the parameter-value element to the desired length of the checksum value.

   For example:

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

4. Save and close the file.

Specifying the Master Person Index Chunk Size

You can specify a number of EUIDs to be allocated at one time to the EUID generator. This allows the generator to assign EUIDs to multiple records without needing to query the sbyn_seq_table database table each time, which improves performance. For detailed information about the chunk size and how EUIDs are allocated, see the *Oracle Healthcare Master Person Index Configuration Reference*.

To Specify the Chunk Size

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the third parameter element, **ChunkSize**, in the EuidGeneratorConfig element.
3. Change the value of the parameter-value element to the desired chunk size.

   For example:

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

4. Save and close the file.

Adding and Deleting Master Person Index EUID Generator Parameters

You cannot add or delete parameters for the default EUID Generator class, but if you create a custom EUID generator class, you can create additional parameters for the new class and delete the existing parameters if they are not used.

To Add EUID Generator Parameters

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **master.xml**.
2. Scroll to the **EuidGeneratorConfig element**.
3. Create a new parameter element inside the parameters element, but outside any existing parameter elements. Add the following elements:
   - **parameter-name** - The name of the parameter.
   - **parameter-type** - The type of parameter. Valid values are java.lang.Long, java.lang.Short, java.lang.Byte, java.lang.String, java.lang.Integer, java.lang.Boolean, java.lang.Double, or java.lang.Float.
   - **parameter-value** - The value of the parameter.

     For example:

```
<parameters>
    <parameter>
      <parameter-name>IdLength</parameter-name>
      <parameter-type>java.lang.Integer</parameter-type>
      <parameter-value>10</parameter-value>
    </parameter>
    <parameter>
      <parameter-name>IncrementBy</parameter-name>
      <parameter-type>java.lang.Integer</parameter-type>
      <parameter-value>5</parameter-value>
   </parameter>
</parameters>
```

4. Save and close the file.

To Delete EUID Generator Parameters

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then **double-click master.xml**.
2. Scroll to the EuidGeneratorConfig element.
3. To delete one parameter, scroll to the parameter element you want to delete and remove all text between and including the **parameter** element.

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

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

4. To remove all parameters, remove all text between and including the parameters element.
5. Save and close the file.

6

Normalization, Standardization, and Phonetic Encoding

This chapter provides information and procedures on how to define, modify and delete normalization and standardization rules; configure the Match and Standardization Engines; and define phonetic encoding for the Master Person Index.

This chapter includes the following sections:

Defining Master Person Index Normalization Rules

Normalization is a part of the standardization process, and is the process of changing non-standard values to a common, standard value. For example, the first name a person uses might not be their given name, but might be a nickname instead. To ensure that a proper match is made between first names, nicknames are normalized based on a configurable list. For example, the common value for "Liz" and "Elizabeth" would be "Elizabeth".

Normalization is defined in mefa.xml. You can define normalization by either using the Configuration Editor or modifying the XML file directly. The changes you make on the Normalization page of the Configuration Editor are reflected in the normalization structures of mefa.xml. The Configuration Editor provides a simplified way of defining normalization.

Perform any of the following tasks to define normalization:

Defining a Master Person Index Field to be Normalized

When you define a field for normalization, you define which field contains the data that needs to be normalized and which field will contain the normalized data. You can

also specify one or more variants to use for normalization. A sample normalization structure for the XML file appears at the end of these instructions.

To Define a Field to be Normalized (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. In the object structure in the left pane, add the field that will contain the normalized value.

   For more information, see Adding a Field to the Master Person Index Object Structure on page 3-5.

3. Click the **Normalization** tab.

   The Normalization page appears.

4. Click **Add**.

   The Normalized Field dialog box appears.

5. Enter or select a value for each of the fields described in Master Person Index Normalization and Standardization Structure Properties on page 6-4.

6. To specify a variant for the type of data being standardized, do the following:

   - In the **Variant Field Name** field, select the field whose value in incoming records will indicate which variant to use.
   - In the Variants section, click **Add**.
   - On the dialog box that appears, enter values in the fields described in Master Person Index Variants Properties on page 6-6.
   - Click **OK**.

     If you selected the multiple domain selector, you can add multiple variants; otherwise, you can add one default variant and one field-defined variant.

7. On the Normalized Field dialog box, click **OK.**

   The new normalization definition appears in the list.

8. On the Configuration Editor toolbar, click **Save**.

To Define a Field to be Normalized (XML Editor)

Before you begin, in object.xml create the field that will contain the new normalized value. For more information, see Adding a Field to the Master Person Index Object Structure on page 3-5.

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.

2. In the structures-to-normalize element, create and name a new group element.

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

3. In the new group element, define the standardization-type and domain-selector attributes (these are described in Master Person Index Normalization and Standardization Structure Properties on page 6-4.

4. If you specified the multiple domain selector for the domain-selector attribute, do the following:
   - In the group element, create a locale-field-name element and a locale-maps element (described in Master Person Index Normalization and Standardization Structure Properties and Master Person Index Variants Properties).
   - For each variant you want to use, define a locale-codes, value, and locale element in the locale-maps element (described in Master Person Index Variants Properties on page 6-6).
5. To specify the source fields to normalize, do the following:
   - Create a new unnormalized-source-fields element in the group element.
   - Create a source-mapping element in the new unnormalized-source-fields element.
   - Define the unnormalized-source-field-name and standardized-object-field-id elements (these are described in Master Person Index Normalization and Standardization Structure Properties on page 6-4).
6. To map the normalized data to destination fields, do the following:
   - Create a new normalization-targets element under the unnormalized-source-fields element that defines the field to map.
   - Create a target-mapping element in the new normalization-targets element.
   - Define the standardized-object-field-id and standardized-target-field-name elements (these are described in Master Person Index Normalization and Standardization Structure Properties on page 6-4).
7. Save and close the file.

Example 6–1 First and Last Name Normalization

```
<structures-to-normalize>
         <group standardization-type="PersonName" domain-selector=
          "com.sun.mdm.index.matching.impl.MultiDomainSelector">
            <locale-field-name>Person.PobCountry</locale-field-name>
            <locale-maps>
              <locale-codes>
                <value>GB</value>
                <locale>UK</locale>
              </locale-codes>
              <locale-codes>
                <value>UNST</value>
                <locale>US</locale>
              </locale-codes>
              <locale-codes>
                <value>Default</value>
                <locale>US</locale>
              </locale-codes>
            </locale-maps>
            <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>
```

Master Person Index Normalization and Standardization Structure Properties

The following table lists and describes the Configuration Editor fields and their corresponding XML elements that define the fields to be normalized or standardized in the master person index application.

You can specify one or more variants for data to be standardized. A variant is a subset of a data type. For example, if the data type is address, variants are defined for addresses from different countries. The rule set for each country is called a variant. For a single variant, you only need to specify the variant if you need to standardize data that is not from the United States. If you are standardizing data from multiple countries, use the multiple domain selector. This requires that one field in the object structure identify which variant to use for each field that will be standardized. For example, the value of the Country field in a system record could be used to tell the standardization engine which variant to use for a particular set of data. If you specified the multiple domain selector in the domain-selector element, you must also define the identifying field and then map the values that can be populated into that field to their corresponding variant.

The following rules apply to the multiple domain selector:

- You can specify a value of **Default** for the identifying field. The corresponding variant is used if the identifying field is blank, contains the value "Default," or contains a value not defined by any of the value elements.
- If a "Default" value is not defined, the system default variant, United States, is used as the default.

For more information about the fields and elements described in the following table, see the *Oracle Healthcare Master Person Index Standardization Engine Reference*.

Configuration Editor Field	XML File Element or Attribute	Description
Data Type	standardization-type	The type of standardization to perform on the source fields. This is specific to the type of data being processed.
Domain Selector	domain-selector	The Java class used by the standardization engine to determine the variant of the data being processed. For the OHMPI Standardization Engine, the following classes can be specified. If no selector is specified, the default is US. The OHMPI Standardization Engine supports Australia, France, Mexico, People's Republic of China (CN), United Kingdom, and United States variants. Possible values for this field are: ■ com.sun.mdm.index.matching.impl. SingleDomainSelectorAU ■ com.sun.mdm.index.matching.impl. SingleDomainSelectorCN ■ 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
Variant Field Name	locale-field-name	The ePath to an identifying field in the object structure that identifies which of the defined variants (element locale-codes) to use. If no field is specified for the OHMPI Standardization Engine, the standardization engine defaults to the United States, regardless of whether any variants are defined. This field must be contained in the object that contains the fields defined for normalization in this structure.
Unnormalized Source	unnormalized-source - field-name	The field that contains the data to be normalized. The field is designated by its ePath (for example, Person.FirstName).
Unnormalized Standardization Component	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 and must correspond to a standardization component defined by that engine.
Normalized Standardization Component	standardized-object-field-id	An identification code that identifies the field that contains the normalized data to the standardization engine. This is specific to the standardization engine in use and must correspond to a standardization component defined by that engine.
Normalized Target	standardized-target-field-name	The field that will store the normalized data. The field is designated by its ePath (for example, Person.Alias[*].StdLastName).

Master Person Index Variants Properties

The following table lists and describes the Configuration Editor fields and XML elements that define a variant for normalization or standardization. In the XML file, each value and locale pair are defined within a locale_codes element. A list of locale_codes elements can be defined in the locale_maps element.

Configuration Editor Field	XML File Element or Attribute	Description
Value	value	A value that indicates to the standardization engine which variant to use to standardize the data. When the value is contained in the Variant Field Name field (or the locale-field-name element), the standardization engine uses the corresponding Variant field (or locale element) to determine the variant. To specify a default variant, enter **Default**.
Variant	locale	A code indicating which variant to use to standardize data when the identifying field value in a transaction matches the corresponding Value field or element. Select one of the following codes. ■ **AU** - Australia ■ **CN** - People's Republic of China ■ **FR** - France ■ **MX** - Mexico ■ **UK** - United Kingdom ■ **US** - United States

Modifying a Master Person Index Normalization Definition

Once you create a normalization definition, you can modify it as needed. Use caution when modifying normalization definitions once a system is in production. This can cause inconsistent match results.

To Modify a Normalization Definition (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **Normalization** tab.

   The Normalization page appears.

3. In the Normalization Mappings list, click the definition you want to modify.
4. Click **Edit**.
5. Do any of the following:
   - Modify any of the fields described in Master Person Index Normalization and Standardization Structure Properties on page 6-4.
   - To modify a variant, select the variant under Variants, and then click **Edit**. Modify either field on the dialog box that appears.
   - To remove a variant, select the variant under Variants, and then click **Remove**. Click **Yes** on the dialog box that appears.
   - Click **OK**.

6. On the Configuration Editor toolbar, click **Save**.

To Modify a Normalization Structure (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click mefa.xml.

   The file opens in the NetBeans XML editor.

2. Scroll to the structures-to-normalize element in the StandardizationConfig element.
3. To modify the normalization type, change the value of the standardization-type attribute.
4. To change the national domain, change the value of the domain-selector element as described in Master Person Index Normalization and Standardization Structure Properties on page 6-4.
5. To modify an existing source field, scroll to the unnormalized-source-fields element in the appropriate group element, and then change the value of any source field elements (these are described in Master Person Index Normalization and Standardization Structure Properties on page 6-4).
6. To modify an existing destination field, scroll to the normalization-targets element in the appropriate group element, and then change the value of any target field elements (these are described in Master Person Index Normalization and Standardization Structure Properties on page 6-4).
7. Save and close the file.

Deleting a Master Person Index Normalization Definition

If a defined normalization structure is not needed, you can delete the normalization structure from the standardization configuration. If no data requires normalization, you can delete all normalization structures. It is not recommended that you delete a normalization definition once a system is in production. This can cause inconsistent match results.

To Delete a Normalization Definition

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. In the Configuration Editor toolbar, click the **Normalization** tab.

   The Normalization page appears.

3. In the Normalization Mappings list, click the definition you want to delete.
4. Click **Remove**.
5. On the Configuration Editor toolbar, click **Save**.

To Delete a Normalization Structure

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **structures-to-normalize element** in the StandardizationConfig element.
3. Do any of the following:
   - To delete an existing normalization structure, delete all text between and including the group element that defines the structure.
   - To specify that no objects require normalization, delete all text between, but not including, the structures-to-normalize element.
4. Save and close the file.

Defining Master Person Index Standardization Rules

If any of the fields against which searching or matching is performed are entered in free-form text format, those fields must be standardized before being sent to the standardization engine. The process of standardization includes reformatting, or parsing, the input data field and then normalizing some of the parsed data to a standard value. For example, street addresses can be parsed into the house number, street name, street type, and so on. The street name and type can then be normalized to their commonly used values. "Ave" might be normalized to "Avenue", "St." to "Street", and so on.

Standardization is defined in mefa.xml. You can define standardization by either using the Configuration Editor or modifying the XML file directly. The changes you make on the Standardization page of the Configuration Editor are reflected in the standardization structures of mefa.xml. The Configuration Editor provides a simplified way of defining standardization.

Perform any of the following tasks to define standardization:

- Defining Master Person Index Fields to be Standardized on page 6-8
- Modifying a Master Person Index Standardization Definition on page 6-12
- Deleting a Master Person Index Standardization Definition on page 6-14

Defining Master Person Index Fields to be Standardized

When you define fields for standardization, you can specify the type of standardization to perform on each field or group of fields, the nationality of the data, and a field that indicates which nationality to use (if you specify more than one). You also specify which fields contain the data that needs to be parsed and normalized, and which fields contain the parsed and normalized data. For each standardization structure, you can specify more than one source field, but they must use the same standardization type. The source fields in one standardization structure are concatenated before being parsed.

A sample standardization structure for the XML file is included at the end of these instructions.

To Define Fields to be Standardized (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.
2. In the object structure in the left pane, create the fields that will contain the parsed components of the new field to be standardized.

For more information, see Adding a Field to the Master Person Index Object Structure on page 3-5.

3. Click the **Standardization** tab.

   The Standardization page appears.

4. Click **Add**.

   The Standardization Type dialog box appears.

5. Enter values for the **Data Type**, **Domain Selector**, and **Variant Field Name** fields (these are described in Master Person Index Normalization and Standardization Structure Properties on page 3-5.

6. To define a variant for the standardization engine to use, do the following:

   - In the Variants section, Click **Add**.
   - On the Variant dialog box, enter values in the fields described in Master Person Index Variants Properties on page 6-6.
   - Click **OK**.

     If you selected the multiple domain selector, you can add multiple variants; otherwise, you can define one default variant and one defined variant.

7. Under Source Fields to be Standardized, click **Add**.

   The Select Source Field(s) dialog box appears.

8. In the left panel, select the field that contains the data that needs to be parsed and normalized, and then click the **right arrow**.

   Note: If the data is contained in more than one field, select all fields that contain the data. For example, a street address might be contained in two fields, such as Street Address and Unit. Both fields should be selected for standardization; they will be concatenated during the standardization process.

9. If you add a field in error, select the field in the Selected Source Field(s) list, and then click the **left arrow**.

10. Click **OK**.

11. For each field in which the parsed and normalized data will be stored, do the following:

    a. On the Standardized Fields dialog box, click **Add** under Target Mappings.

       The Target Mapping dialog box appears.

    b. In the **Select Target** field, select the name of a field that will contain standardized data.

    c. In the Available Standardization Components list, select the ID associated with the field, and then click **Add** between the left and right panels.

    d. To change the priority of a component in the Selected Standardization Components list, select the component and then click **Move Up** or **Move Down**.

    e. If you add a component in error, select the component in the Selected Standardization Components list, and then click **Remove**.

f. Click **OK**.

Note: For more information about standardization components and the fields to which they pertain, see the *Oracle Healthcare Master Person Index Standardization Engine Reference*.

---

12. Click **OK** on the Standardization Type dialog box.

    The new standardization definition appears in the list.

13. On the Configuration Editor toolbar, click **Save**.

To Define Fields to be Standardized (XML Editor)

Before you begin, in object.xml create the fields that will contain the parsed components of the field to be standardized. For more information, see Adding a Field to the Master Person Index Object Structure on page 3-5.

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.

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

3. Create a new group element in the free-form-texts-to-standardize element, and then define the standardization-type and domain-selector attributes (these are described in Master Person Index Normalization and Standardization Structure Properties on page 6-4).

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

4. If you specified the multiple domain selector for the domain-selector attribute, in the group element create a **locale-field-name element** and a **locale-maps element**, and define the elements described in Master Person Index Variants Properties on page 6-6.

5. To specify the source fields to standardize, do the following:

   - If it does not currently exist, create an unstandardized-source-fields element in the appropriate group element (each group element can only include one unstandardized-source-fields element).
   - For each field standardized by the specified standardization type, create and name a new unstandardized-source-field-name element in the new unstandardized-source-fields element.

   Note: If more than one source field is defined, the fields are concatenated prior to standardization with a pipe (|) between them for the OHMPI Standardization Engine. If you want the fields to be processed separately, you need to create two standardization structures. Source fields are designated by their ePaths.

   ---

6. To specify the destination fields for the standardized data, do the following:

- In the group element for which destination fields need to be defined, create a standardization-targets element after the unstandardized-source-fields element.
- In the new element, create a target-mapping element for each destination field, and then define the last two elements described in Master Person Index Standardization Source and Target Field Elements on page 6-12.

7. Save and close the file.

Example 6–2 Address Standardization Structure

```
<group standardization-type="Address" domain-selector=
 "com.sun.mdm.index.matching.impl.MultiDomainSelector">
  <locale-field-name>Person.Address[*].CountryCode
  </locale-field-name>
  <locale-maps>
     <locale-codes>
         <value>GB</value>
         <locale>UK</locale>
      </locale-codes>
      <locale-codes>
         <value>UNST</value>
         <locale>US</locale>
      </locale-codes>
      <locale-codes>
         <value>AU</value>
         <locale>AU</locale>
      </locale-codes>
      <locale-codes>
         <value>Default</value>
         <locale>AU</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>
   </standardization-targets>
</group>
```

Master Person Index Standardization Source and Target Field Elements

The following table lists and describes the XML elements that define the source and target fields for standardization. The data from the source fields is standardized, and the standardized values are stored in the target fields.

XML File Element or Attribute	Description
unstandardized-source-field-name	The field or fields that contain the data to be standardized. The field is designated by its ePath (for example, Person.FirstName).
standardized-object-field-id	A code that identifies the standardized component from the source field to store in the target field. This is specific to the standardization engine in use and must correspond to a standardization component defined by that engine. For more information, see the *Oracle Healthcare Master Person Index Standardization Engine Reference*.
standardized-target-field-name	The field that stores the standardized data. You can have multiple target fields, depending on how much of the standardized data you want to store. The fields are designated by their ePaths (for example, Person.Alias[*].StdLastName).

Modifying a Master Person Index Standardization Definition

You can modify an existing standardization definition. Use caution when modifying standardization after a system is in production because it can cause inconsistent matching results.

To Modify a Standardization Definition (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **Standardization** tab.

   The Standardization page appears.

3. In the Standardization Types list, select the definition you want to modify, and then click **Edit**.

   The Standardization Type dialog box appears.

4. Do any of the following:

   - Modify any of the fields or perform any of the functions described in Defining Master Person Index Fields to be Standardized on page 6-8.
   - To modify a variant, select the code under Variants, and then click **Edit**. Modify either field on the dialog that appears.
   - To remove a variant, select the code under Variants, and then click **Remove**. Click **Yes** on the dialog box that appears.
   - To remove a source field, select the field under Source fields to be standardized, and then click **Remove**. Click **Yes** on the dialog box that appears.

   Note: There must be at least one field in this list.

- To edit a target field, select the field in the Specifying Target Mappings list and then click **Edit**.

  Note: You can select new components, move selected components up and down in priority, and remove components.

- To delete a target field, select the field in the Specifying Target Mappings list, and then click **Remove**. Click **Yes** on the dialog box that appears.

5. When you are done making changes, click **OK** on the Standardization Type dialog box.
6. On the Configuration Editor toolbar, click **Save**.

To Modify a Standardization Definition (XML Editor)

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.
2. Scroll to the **structures-to-normalize element** in the StandardizationConfig element.
3. To modify the standardization type, change the value of the standardization-type attribute.
4. To change the variant, change the value of the domain-selector element (described in Master Person Index Normalization and Standardization Structure Properties on page 6-4).
5. To modify an existing source field, scroll to the appropriate group element, and then change the value of the unstandardized-source-field-name element to the ePath of the new field.
6. To modify an existing destination field, scroll to the **target-mapping element** in the standardization-targets section, and then change the value of either target mapping element (these are the last two elements described in Master Person Index Standardization Source and Target Field Elements on page 6-12).
7. To remove an existing source field, delete all text between and including the unstandardized-source-field-name element that defines the field.

   Note: If no fields require standardization in a defined standardization structure, delete the entire structure as described in Deleting a Master Person Index Standardization Definition on page 6-14.

8. To remove an existing destination field, delete all text between and including the target-mapping tags that define the field.

   Note: Each standardization structure must have at least one destination field defined for standardized data. If a structure does not contain any fields that need to be standardized, you can delete the entire structure, as described in Deleting a Master Person Index Standardization Definition on page 6-14.

9. Save and close the file.

Deleting a Master Person Index Standardization Definition

You can delete an existing standardization definition. It is not recommended that a standardization definition be deleted after a system is in production since this can cause inconsistent matching results.

To Delete a Standardization Definition (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **Standardization** tab.

   The Standardization page appears.

3. In the Standardization Types list, select the definition you want to delete.
4. Click **Remove**.
5. Click **Yes** on the dialog box that appears.
6. On the Configuration Editor toolbar, click **Save**.

To Delete a Standardization Definition (XML Editor)

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **free-form-texts-to-standardize element** in the StandardizationConfig element.
3. Do any of the following:
   - To delete an existing standardization structure, delete all text between and including the group element that defines the structure.

     Using the example below, to delete the Address object, delete all boldface text.

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

   - To specify that no fields require standardization, delete all text between, but not including, the free-form-texts-to-standardize element.

     This deletes all standardization structures.

4. Save and close the file.

Defining Phonetic Encoding for the Master Person Index

Oracle Healthcare Master Person Index provides configurable phonetic encoding capabilities. Phonetic encoding is a part of the standardization process, and is the process of changing the value of a data field to its phonetic equivalent. It is used to retrieve records with similar field values from the database for matching. You can specify which fields are phonetically encoded before matching and how they are encoded. There are several different encoders you can use for this purpose. This is most commonly done for first names and street names.

Phonetic encoding is defined in mefa.xml. You can define phonetic encoding by either using the Configuration Editor or modifying the XML file directly. The changes you make on the Phoneticized Field page of the Configuration Editor are reflected in the phonetic encoding structures and the match service section of mefa.xml. The Configuration Editor provides a simplified way of defining phonetic encoding.

Perform any of the following tasks to define phonetic encoding rules:

- Defining Master Person Index Fields for Phonetic Encoding on page 6-15
- Modifying a Master Index Phonetic Encoding Definition on page 6-17
- Deleting a Master Person Index Phonetic Encoding Definition on page 6-18
- Defining a Master Person Index Phonetic Encoder on page 6-19
- Modifying a Master Person Index Phonetic Encoder on page 6-20
- Deleting a Master Person Index Phonetic Encoder on page 6-21

Defining Master Person Index Fields for Phonetic Encoding

You can specify the fields you want to be phonetically encoded, the fields that store the encoded values, and the type of phonetic encoder to use for each field, such as NYSIIS or Soundex. A sample phonetic encoding structure for the XML file is included at the end of these instructions.

To Define a Field for Phonetic Encoding (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. In the object structure in the left pane, create the field that will contain the phonetic value of the field to be encoded.

   For more information, see Adding a Field to the Master Person Index Object Structure on page 3-5.

3. Click the **Phoneticized Fields** tab.

   The Phoneticized Field page appears.

4. In the Phoneticized Fields section, click **Add**.

   The Phoneticized Field dialog box appears.

5. Select values for the fields described in Master Person Index Phonetic Encoding Fields and Elements on page 6-16.

6. Click **OK**.

   The phonetic encoding definition is added to the Phoneticized Fields list.

7. On the Configuration Editor toolbar, click **Save**.

To Define a Field for Phonetic Encoding (XML Editor)

In object.xml, create the field that will contain the phonetic value. For more information, see Adding an Object to the Master Person Index Object Structure on page 3-2.

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **phoneticize-fields element** in the PhoneticEncodersConfig element.
3. Create a new phoneticize-field element within the phoneticize-fields element.
4. In the new phoneticize-field element, create and define the elements described in Master Person Index Phonetic Encoding Fields and Elements on page 6-16.
5. Save and close the file.

Example 6–3 Phonetic Encoding Structure

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

Master Person Index Phonetic Encoding Fields and Elements

The following table lists and describes the Configuration Editor fields and XML file elements used to define how fields will be phonetically encoded.

Configuration Editor Field	XML File Element	Description
Unphoneticized Source	unphoneticized-source-field-name	The ePath of the source field in the system object that contains the data to be phonetically encoded (for example, Person.Address[*].StreetName). **Note**: This can refer to the original field or to a standardized or normalized field.
Phoneticized Target	phoneticized-target-field-name	The ePath of the field in the system object that will store the phonetically encoded value of the source field.

Configuration Editor Field	XML File Element	Description
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, but could be used with a custom standardization engine.	
Encoder	encoding-type	The phonetic encoder to use for this field. This must correspond to an encoder defined in the Encoders section on the Configuration Editor (or the PhoneticEncodersConfig element of the XML file).

Modifying a Master Index Phonetic Encoding Definition

Once you create a phonetic encoding definition, you can modify it as needed. Use caution when modifying definitions once a system is in production. This can cause inconsistent match results.

To Modify a Phonetic Encoding Definition (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **Phoneticized Fields** tab.

   The Phoneticized Field page appears.

3. In the Phoneticized Fields section, select the phonetic encoding definition you want to modify, and then click **Edit**.

   The Phoneticized Field dialog box appears.

4. Modify any of the fields listed in Master Person Index Phonetic Encoding Fields and Elements on page 6-16, and then click **OK**.

5. On the Configuration Editor toolbar, click **Save**.

To Modify a Phonetic Encoding Definition (XML Editor)

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.

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

3. In the phoneticize-field element that defines the phonetic encoding you want to modify, change the value of any of the elements described in Master Person Index Phonetic Encoding Fields and Elements on page 6-16.

4. Save and close the file.

Deleting a Master Person Index Phonetic Encoding Definition

Once you create a phonetic encoding definition, you can delete it as needed. Use caution when deleting definitions once a system is in production. This can cause inconsistent match results.

To Delete a Phonetic Encoding Definition (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **Phoneticized Fields** tab.

   The Phoneticized Field page appears.

3. In the Phoneticized Fields section, select the phonetic encoding definition you want to delete.

4. Click **Remove**.

5. Click **Yes** on the dialog that appears.

6. On the Configuration Editor toolbar, click **Save**.

To Delete a Phonetic Encoding Definition (XML Editor)

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.

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

3. Do any of the following:

   - To delete a field currently specified for phonetic conversion, delete all text between and including the phoneticize-field element that defines the field.

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

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

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

4. Save and close the file.

Defining a Master Person Index Phonetic Encoder

Each type of phonetic encoder provided with Oracle Healthcare Master Person Index is defined in the PhoneticEncodersConfig section of mefa.xml. They are listed in the Encoders section of the Configuration Editor.

To Define a Phonetic Encoder (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **Phoneticized Fields** tab.

   The Phoneticized Field page appears.

3. In the Encoders section, click **Add**.

   The Phonetic Encoder dialog box appears.

4. In the **Encoder** field, enter a descriptive name for the encoder.

5. In the **Implementation Class** field, enter the fully qualified Java path of the class to use for the encoder.

   Note: For more information about the encoder class paths, see Master Person Index Encoder Elements and Types on page 6-19.

6. Click **OK**.

   The phonetic encoder is added to the Encoders list.

7. On the Configuration Editor toolbar, click **Save**.

To Define a Phonetic Encoder (XML Editor)

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **PhoneticEncodersConfig section**.

3. Create a new encoder element, and then define the elements described in Master Person Index Encoder Elements and Types on page 6-19.

4. Save and close the file.

Master Person Index Encoder Elements and Types

The following table lists and describes the elements that configure the phonetic encoders used by the master person index application.

Element	Description
encoding-type	The name of the phonetic encoder, such as NYSIIS, Soundex, or Metaphone. See the following table for a list of default encoders for the OHMPI Standardization Engine.
encoder-implementation- class	The fully qualified name of the Java class that determines the behavior of the phonetic encoder. See the following table for a complete list of default classes for the OHMPI Standardization Engine.

The following table lists the phonetic encoders supported by the Master Index Standardization Engine along with the names of their default classes.

Encoding Type	class-name
NYSIIS	com.sun.mdm.index.phonetic.impl.Nysiis
Soundex	com.sun.mdm.index.phonetic.impl.Soundex
Metaphone	com.sun.mdm.index.phonetic.impl.Metaphone
Double Metaphone	com.sun.mdm.index.phonetic.impl.DoubleMetaphone
Refined Soundex	com.sun.mdm.index.phonetic.impl.RefinedSoundex
French Soundex	com.sun.mdm.index.phonetic.impl.SoundexFR
HanYuPinYin	com.sun.mdm.index.phonetic.impl.HanYuPinYin
HanYuPinYin_ NoTone	com.sun.mdm.index.phonetic.impl.HanYuPinYinNoTone
HanYuPinYin_ LastName	com.sun.mdm.index.phonetic.impl.HanYuPinYinLastName
HanYuPinYin_ LastName_NoTone	com.sun.mdm.index.phonetic.impl.HanYuPinYinLastNameNoTone

Modifying a Master Person Index Phonetic Encoder

Once you define a phonetic encoder, you can change the implementation class to use for the encoder. Use caution when modifying definitions once a system is in production. This can cause inconsistent match results.

To Modify a Phonetic Encoder (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Click the **Phoneticized Fields** tab.

   The Phoneticized Field page appears.

3. In the Encoders section, select the encoder you want to modify and then click **Edit**.

   The Phonetic Encoder dialog box appears.

4. Change the implementation class, and then click **OK**.

5. On the Configuration Editor toolbar, click **Save**.

To Modify a Phonetic Encoder (XML Editor)

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **PhoneticEncodersConfig section**.

3. Modify the value of any of the elements in the encoder element you want to modify (for more information, see Master Person Index Encoder Elements and Types on page 6-19.

4. Save and close the file.

Deleting a Master Person Index Phonetic Encoder

Once you define a phonetic encoder, you can delete it if needed. Use caution when deleting encoders once a system is in production. This can cause inconsistent match results. If you delete an encoder that is referenced by a phonetic encoding definition, make sure to modify that definition by referencing an existing encoder.

To Delete a Phonetic Encoder (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. In the Configuration Editor toolbar, click the **Phoneticized Fields** tab.

   The Phoneticized Field page appears.

3. In the Encoders section, select the encoder you want to delete.
4. Click **Remove**.
5. Click **Yes** on the dialog box that appears.
6. On the Configuration Editor toolbar, click **Save**.

To Delete a Phonetic Encoder (XML Editor)

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **PhoneticEncodersConfig section**.
3. Delete the text between and including the encoder element that defines the encoder you want to remove.
4. Save and close the file.

Defining the Master Person Index Match String

The match string defines the fields that are passed to the match engine for probabilistic weighting. By default, the fields defined for matching are the fields you specified for matching in the wizard or Configuration Editor. You can modify and delete fields in the match string if necessary. At least one field must be defined in the match string or no weights will be generated.

If you do modify the match string, you might need to make corresponding changes to the match engine configuration files as well. For more information about modifying these files, see the *Oracle Healthcare Master Person Index Match Engine Reference*.

Perform either of the following tasks to configure the match string.

- Creating the Master Person Index Match String on page 6-22
- Modifying the Master Person Index Match String on page 6-23

You can further configure the match string by defining exclusion lists that filter out unwanted values from the match process. For more information, see Filtering Default Values From Master Person Index Processes on page 8-1.

Creating the Master Person Index Match String

A default match string is predefined based on the match type information you specified in the wizard. If no match types were defined using the wizard, the structure of the match string is still defined but with no fields. You can use normalized or phonetically encoded fields for the match string.

To Create the Match String (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.

2. Expand the object structure so all fields are visible.

3. To add a field to the match string, click on the field name and then select a value for the Match Type field on the Properties page.

   Note: The match types you can use are listed in the first column of matchConfigFile.cfg. For more information about OHMPI Match Engine match types, see the *Oracle Healthcare Master Person Index Match Engine Reference*.

4. Perform the previous step for each field in the match string.

5. On the Configuration Editor toolbar, click **Save.**

6. To remove unwanted or invalid field values from the match process, see Filtering Default Values From Master Person Index Processes on page 8-1.

To Create the Match String (XML Editor)

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.

2. In the MatchingConfig element, scroll to the **match-columns element** in the match-system-object element.

3. To add a field to the match string, do the following:

   1. In the match-columns element, create a new match-column element.

   2. In the new match-column element, create and define a column-name element.

      Enter the fully qualified field name of the field on which to match (for example, Enterprise.SystemSBR.Person.Address.City).

   3. Following the column-name element, create and define a match-type element.

      Enter an ID that identifies the field to the match engine. For the OHMPI Match Engine, this value must correspond to a defined match type.

      For example:

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

```
            <match-type>StreetName</match-type>
          </match-column>
        </match-columns>
      </match-system-object>
```

4. Repeat the previous step for each field to add to the match string.
5. Save and close the file.
6. To remove unwanted or invalid field values from the match process, see Filtering Default Values From Master Person Index Processes on page 8-1.

Modifying the Master Person Index Match String

Once you define a match string, you can modify or delete information about the fields in the match string as necessary. This should only be done prior to moving to production. Otherwise, unexpected matching results might occur. For more information about OHMPI Match Engine match types and field IDs, see the *Oracle Healthcare Master Person Index Match Engine Reference*.

To Modify the Match String (Configuration Editor)

1. In the Projects window, right-click the **Configuration node** in the project you want to modify, and then click **Edit**.

   The Configuration Editor appears.
2. Expand the object structure so all fields are visible.
3. To add a field to the match string, click the field name and then select a value for the Match Type field on the Properties page.

   The field is added to the match string.
4. To modify the match type specified for a field, click the name of the field defined for matching and then select a new value for the Match Type field on the Properties page.
5. To remove a field from the match string, click the name of the field defined for matching and then select **None** for the Match Type field on the Properties page.
6. On the Configuration Editor toolbar, click **Save**.
7. To define or modify exclusion lists of values to be removed from the match process, see Filtering Default Values From Master Person Index Processes on page 8-1.

To Modify the Match String (XML Editor)

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.
2. Scroll to the **MatchingConfig element**, and then scroll to the match-system-object element.
3. To add a field to the match string, do the following:
   - In the match-columns element, create a new match-column element.
   - In the new match-column element, create and define a column-name element.

     Enter the fully qualified field name of the field on which to match (for example, `Enterprise.SystemSBR.Person.Address.City`).

- Following the column-name element, create and define a match-type element.

  Enter an ID that identifies the field to the match engine. For the OHMPI Match Engine, this value must correspond to a defined match type.

  For example:

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

4. To change a field used in the match string, change the value of the column-name element.

   Enter the fully qualified field name of the new field (for example, `Enterprise.SystemSBR.Person.FirstName`).

5. To change the type of matching to perform for a field, change the value of the match-type element.

   Enter an ID that identifies the field to the match engine. For the OHMPI Match Engine, this value must correspond to a defined match type.

6. To delete a field from the match string, delete all text between and including the match-column element defining the field you want to delete.

   Using the example below, to delete the HouseNo field from the match string, delete the boldface text.

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

7. Save and close the file.

8. To define or modify exclusion lists of values to be removed from the match process, see Filtering Default Values From Master Person Index Processes on page 8-1.

Defining How Master Person Index Query Blocks are Processed

The block picker and pass controller define how query blocks are processed for matching. Default components are defined by Oracle Healthcare Master Person Index,

but you can create your own Java classes to define custom versions of these components.

The block picker determines the blocking strategy to use for each match pass. Blocking strategies define how the queries are created that check the database for a subset of the records to be used for matching. The default Block Picker has access to the match results from previous match passes, as well as lists of applicable blocking definitions that have been executed and of those that have not. The default Block Picker class is com.sun.mdm.index.matching.impl.PickAllBlocksAtOnce, which selects all blocks during the first pass.

The pass controller determines whether to continue processing the defined blocks. The matching process can be executed in multiple stages. Each query block in the blocking query is executed in a separate match pass. After a block is evaluated, the pass controller determines if the results found are sufficient or if the query should continue by performing another match pass. The default pass controller is com.sun.mdm.index.matching.impl.PassAllBlocks. This class instructs the match engine to continue calculating match weights until all applicable block definitions have been processed.

These components can only be configured by modifying the XML file directly.

To Specify the Block Picker

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **block-picker element** in the MatchingConfig section.

3. In the class-name element, specify the Java class for the block picker you want to use, using the fully qualified class name.

   For example:

```
<block-picker>
   <class-name>com.sun.mdm.index.matching.impl.PickAllBlocksAtOnce
   </class-name>
</block-picker>
```

4. Save and close the file.

To Specify the Pass Controller Class

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **pass-controller element** in the MatchingConfig section.

3. In the class-name element, specify the Java class for the Pass Controller you want to use, using the fully qualified class name.

   For example:

```
<pass-controller>
   <class-name>com.sun.mdm.index.matching.impl.PassAllBlocks
   </class-name>
</pass-controller>
```

4. Save and close the file.

Configuring the Standardization Engine

You can configure the standardization engine by specifying the standardization engine to use, configuring the files that define data standardization, and plugging in custom standardization and matching rules. You only need to specify the standardization engine to use if you are using an engine other than the OHMPI Standardization Engine.

Perform any of these steps to configure the standardization engine:

- Specifying a Standardization Engine for the Master Person Index on page 6-26
- Modifying Master Index Standardization Files on page 6-27
- Importing Standardization Data Types and Variants on page 6-27
- Deleting a Standardization Variant or Data Type on page 6-28

Specifying a Standardization Engine for the Master Person Index

Oracle Healthcare Master Person Index can support standardization engines from different vendors depending on the adapter configured to communicate with the engine. Default classes are provided for using the OHMPI Standardization Engine. You can implement a custom standardization engine along with customized adapters. The standardization engine configuration is defined by standardizer-api and standardizer-config elements.

> **Note:** The default adapters for the OHMPI Standardization Engine are com.sun.mdm.index.matching.adapter.SbmeStandardizerAdapter and com.sun.mdm.index.matching.adapter.SbmeStandardizerAdapterConfig.

To Specify the Standardization Engine

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **mefa.xml**.

   The file opens in the NetBeans XML editor.

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

3. Specify the Java class for the standardization adapter to use, using the fully qualified class name as shown below.

```
<standardizer-api>
   <class-name>
    com.sun.mdm.index.matching.adapter.MyStandardizerAdapter
   </class-name>
</standardizer-api>
```

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

```
<standardizer-config>
   <class-name>
     com.sun.mdm.index.matching.adapter.SbmeStandardizerAdapterConfig
   </class-name>
</standardizer-config>
```

5. Save and close the file.

Modifying Master Index Standardization Files

You can fine-tune the standardization process by modifying the standardization files. For example, you can insert additional names or terms into the normalization or lexicon files, such as giventNames.txt and givenNameNormalizatin.txt. Depending on your data requirements, you might need to modify additional standardization files. Some of the patterns files (most notably the address patterns files) are very complex and should only be modified by personnel who thoroughly understand the defined patterns and tokens. If you modify standardization files, make sure you modify them for each variant specified in mefa.xml.

You can modify the data configuration files (lexicon and normalization files), and you can also modify the process configuration files that define the data types, variants, and how data is standardized. The process files are more complex, and should only be modified by one who is familiar with standardization concepts and with the OHMPI Standardization Engine. Instructions for modifying these files are not included here. For information about these files, see the *Oracle Healthcare Master Person Index Standardization Engine Reference*.

To Modify Standardization Data Configuration Files

1. In the Projects window, expand the master person index project to configure and then expand Standardization Engine.
2. Expand instance, expand the variant to modify, and then expand resources.
3. Open the file you want to modify in the NetBeans text editor.
4. Modify the file in accordance with the information presented for each data type in *Oracle Healthcare Master Person Index Standardization Engine Reference*.
5. Save and close the file.

Importing Standardization Data Types and Variants

The OHMPI Standardization Engine is based on a very flexible framework that allows you to define new data types and variants so you can standardize any type of data in a custom manner. You can create new data types and variants based on the finite state machine and new variants for the existing rules-based data types. You need to import the data type or variant package into NetBeans to make it available to all master person index applications or only the current one.

This section only describes importing custom data types and variants after they have been created. For information about creating a custom data type or variant, see the *Oracle Healthcare Master Person Index Standardization Engine Reference*.

To Import a Data Type or Variant

1. In the Projects window, expand the **main master person index project**.
2. Right-click **Standardization Engine**, and select **Import Standardization Plug-in**.
3. In the dialog box that appears, navigate to the location of the plug-in package.
4. Select the file containing the plug-in, and then click **Open**.
5. Do one of the following:
   - To import the plug-in and make it available to all future master person index applications, click **Yes**.
   - To import the plug-in and make it only available to the current master person index application, click **No**.

The data type or variant is imported into the Standardization Engine node. Data types add folders just beneath the Standardization Engine node; variants add folders under the appropriate data type (as specified in the variant package).

6. In the Standardization Engine node, navigate to the new data type or variant you added and verify that all of the required files are there.

Deleting a Standardization Variant or Data Type

If you add a data type or variant to a master person index application in error, you can remove it from the Standardization Engine node. You can also delete any of the existing data types or variants if they are not in use.

Caution: Be careful when removing variants or data types; this action cannot be undone.

To Delete a Variant or Data Type

1. Back up the source files for the data type or variant in case you need them at a later time.

   The default data types and variants are stored in NetBeansHome/soa2/modules/ext/mdm/standardizer/deployment.

2. In the Project window, navigate to the **Standardization Engine node** in the master person index project and then to the data type or variant you want to remove.

3. Right-click the folder containing the files to remove, and then select **Delete**.

   A confirmation dialog appears.

4. Click **Yes**.

   The data type or variant is removed from the project.

7

Master Person Index Survivor Calculator

This chapter provides information and procedures on how to define and configure the Survivor Calculator.

This chapter includes the following section:

Defining the Master Person Index Survivor Calculator

The survivor calculator is configured in update.xml and defines how field values are populated into the single best record, along with the type of survivor strategy to use for each field. If no strategy is defined for a field, the default survivor strategy is used to populate that field in the SBR. Before you begin to define the survivor calculator, make sure you know which fields to include in the SBR and how those fields should be updated. Typically, all but phonetic and standardized fields are included here.

You can further configure the survivor calculator by defining exclusion lists to filter out unwanted values from the SBRs. For more information, see Filtering Default Values From Master Person Index Processes on page 8-1.

The survivor calculator can only be configured by modifying the XML file directly. Perform the following tasks to configure the survivor calculator.

Specifying the Master Person Index Survivor Helper

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

To Specify the Survivor Helper

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **update.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **helper-class element** in the SurvivorHelperConfig element.

3. Change the value of the helper-class element to the fully qualified name of new helper class.

   For example:

   ```
   <helper-class>com.sun.mdm.index.survivor.impl.MySurvivorHelper
   </helper-class>
   ```

4. Save and close the file.

Specifying a Master Person Index Default Survivor Strategy

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

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

To Specify a Default Survivor Strategy

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **update.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **default-survivor-strategy element** in the SurvivorHelperConfig element.

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

   For example:

   ```
   <default-survivor-strategy>
      <strategy-class>com.sun.mdm.index.survivor.impl.MySurvivorStrategy
      </strategy-class>
   </default-survivor-strategy>
   ```

4. Configure the strategy parameters, as described in Configuring the Default Survivor Strategy on page 7-2.

5. Save and close the file.

Configuring the Default Survivor Strategy

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

To Configure the Default Survivor Strategy

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **update.xml**.

   The file opens in the NetBeans XML editor.

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

3. Do any of the following:

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

     For example:

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

   - To add a new parameter, create a new parameter element within the parameters element, and then define the parameter elements described in Master Person Index Default Survivor Strategy Parameter Elements on page 7-3.
   - To delete a parameter, scroll to the parameters element, and then delete all text between and including the parameter element that define the parameter.
   - To delete all parameters, delete the all text between and including the parameters element.

4. Save and close the file.

Master Person Index Default Survivor Strategy Parameter Elements

The following table lists and describes the elements that configure the parameters for the default survivor strategy in update.xml.

Element	Value
description	This is an optional element that briefly describes the parameter. Note that there are no parameters to define for the UnionSurvivorStrategy.
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.

Element	Value
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).

Defining the Master Person Index Single Best Record Structure

In order for a field to be populated in the SBR, that field must be defined in the candidate field list of the survivor helper. By default, all the fields that were specified in the wizard are also defined here. Any candidate fields defined for the SBR must also be defined in object.xml. If you add a field to the object definition, you should also add the field here.

The SBR can only be defined by modifying the XML file directly.

To Specify a Candidate Field

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **update.xml**.

   The file opens in the NetBeans XML editor.

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

3. Do any of the following:

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

     For example:

     ```
     <candidate-definitions>
           <candidate-field name="Person.LastName"/>
           <candidate-field name="Person.DateOfBirth"/>
        <candidate-definitions>
     ```

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

   - If any of the updated fields do not use the default strategy, define a strategy for those fields, as described in Defining a Master Person Index Survivor Strategy for a Field or Object on page 7-5.

4. Save and close the file.

5. To define or modify exclusion lists of values to be excluded from the SBR, see Filtering Default Values From Master Person Index Processes on page 8-1.

Deleting Candidate Fields

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

To Delete a Candidate Field

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **update.xml**.

   The file opens in the NetBeans XML editor.

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

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

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

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

   Note: Do not delete all candidate fields; at a minimum, the match fields must be defined.

4. Save and close the file.

Defining a Master Person Index Survivor Strategy for a Field or Object

To use a strategy for a specific field other than the strategy defined in the default-survivor-strategy element, you need to specify the new strategy for the appropriate candidate-field element. A candidate-field element can represent a field or child object. You do not need to specify a strategy for any fields that use the default survivor strategy.

To Define a Survivor Strategy for a Field

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **update.xml**.

   The file opens in the NetBeans XML editor.

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

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

   For example:

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

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

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

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

Note: To specify the default survivor strategy for a field, make sure the corresponding candidate-field element does not contain a survivor-strategy element. If you implement a custom strategy class, that class must be defined as a custom plug-in.

5. Save and close the file.
6. To define or modify exclusion lists of values to be excluded from the SBR, see Filtering Default Values From Master Person Index Processes on page 8-1.

Defining Master Person Index Custom Weighted Strategies

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

Configuring the weighted calculator involves the following tasks.

- Defining Custom Weighted Strategies on page 7-6
- Configuring Weighted Strategies on page 7-7
- Modifying Weighted Calculator Parameters on page 7-8
- Deleting Weighted Calculator Parameters on page 7-8

Defining Custom Weighted Strategies

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

To Define Custom Weighted Calculators

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **update.xml**.

   The file opens in the NetBeans XML editor.
2. Scroll to the **WeightedCalculator element**.
3. Add and name a new candidate-field element in the WeightedCalculator.

   For example:

```
<WeightedCalculator module-name="WeightedSurvivorCalculator"
 parser-class=
```

```
 "com.sun.mdm.index.configurator.impl.WeightedCalculatorConfig">
   <candidate-field name="Person.DOB">
   </candidate-field>
```

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

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

5. For each new parameter element, define the elements listed in Master Person Index Weighted Calculator Parameter Elements on page 7-9.

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

6. Save and close the file.

Configuring Weighted Strategies

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

To Configure Weighted Strategies

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **update.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **default-parameters element** in the WeightedCalculator element.

3. Create a new parameter element within the default-parameters element, but outside any existing parameter elements.

4. In the new parameter element, define the elements listed in Master Person Index Weighted Calculator Parameter Elements on page 7-9.

   For example:

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

```
        <utility>75.0</utility>
      </parameter>
</default-parameters>
```

5. Save and close the file.

Modifying Weighted Calculator Parameters

Once a candidate field is specified and custom weighted calculators are defined for the field, you can modify the parameters. You can also modify any existing default weighted calculator parameters.

To Modify Weighted Calculator Parameters

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **update.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **WeightedCalculator element**.

3. Do any of the following:

   - To modify a custom weighted calculator parameter, scroll to the **candidate-field element** naming the field to modify.
   - To modify a default weighted calculator parameter, scroll to the **default-parameters element**.

4. Modify the value of any of the elements listed in Master Person Index Weighted Calculator Parameter Elements on page 7-9.

   For example:

```
<parameter>
   <quality>SourceSystem</quality>
   <preference>DDI</preference>
   <utility>60.0</utility>
</parameter>
```

5. Save and close the file.

Deleting Weighted Calculator Parameters

Once default and custom parameters are defined, they can be deleted if necessary. If a candidate field is defined for custom weighted calculations, you can specify that the field use the default weighted calculator instead by removing the entire field from the candidate-fields list.

To Delete Weighted Calculator Parameters

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **update.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **WeightedCalculator element** and then to the candidate-field element identifying the field you want to delete.

3. Do any of the following:

   - To delete a custom weighted calculator parameter, scroll to the **candidate-field element** naming the field to modify.

- To delete a default weighted calculator parameter, scroll to the **default-parameters element**.

4. To delete an existing parameter, scroll to the parameter element you want to delete, and then remove all text between and including the parameter element.

   For example, to delete the SourceSystem parameter below, delete the boldface text.

```
<parameter>
   <quality>SourceSystem</quality>
   <preference>CDI</preference>
   <utility>80.0</utility>
</parameter>
<parameter>
   <quality>MostRecentModified</quality>
   <utility>75.0</utility>
</parameter>
```

   Note: At least one parameter must be defined for the default-parameter element; you cannot delete all parameters from this section. You cannot delete all parameters from a candidate field, but you can delete the entire candidate field (see below for more information).

5. To delete a field from the candidate field list, delete all text between and including the candidate-field tags for the field you want to delete.

   Using the following example, to delete the Person.DOB candidate field from the custom calculator, delete all the text below.

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

6. Save and close the file.

Master Person Index Weighted Calculator Parameter Elements

The following table lists and describes the elements that configure the parameters for the weighted calculator in update.xml.

Element	Description
quality	The type of weighted calculation to perform, such as: ■ SourceSystem ■ SystemAgreement ■ MostRecentModified For more information about these qualities, see the *Oracle Healthcare Master Person Index Configuration Reference*.
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.

8

Processes, Update Policies, and Field Validations

This chapter provides information and procedures to filter default values from processes, configure update policies, and define custom field validations.

This chapter includes the following sections:

Filtering Default Values From Master Person Index Processes

You might want to prevent certain default values from being used in the match process or from being populated into the single best record (SBR). For the blocking query, default values result in a large number of non-matching records being returned for a match query. For the match process, default values result in inaccurately high match weights. Finally, you do not want invalid values to appear in the SBR when it is possible to remove them.

The master person index application provides a configuration file, filter.xml, in which you can define filtering rules. You can specify fields and values to filter directly in this file, and you can create exclusion lists in a flat file. Exclusion lists contain all the values to filter for a specific field.

To Filter Default Values From the SBR, Blocking Query, or Match Process

1. To use flat files to define values to filter, create a flat file for each field to be filtered.
2. In this file, list each value to exclude for each filter definition you will create and separate each value by a delimiter.
3. In the Projects window, expand the **master person index application** and then expand **Filter**.
4. Open **filter.xml**.
5. For each field to be filtered, define the elements and attributes described in Filter Definition Elements on page 8-2.
6. Save and close the file.

Example 8–1 Filter Definitions

```
<exclusion-List module-name="ExclusionFilter"
parser-class="com.sun.mdm.index.configurator.impl.ExclusionFilterConfig">
  <field sbr="true" matching="false" blocking="false">
    <name>Person.FirstName</name>
    <value
      <field-value>Baby</field-value>
      <field-value>Baby Boy</field-value>
      <field-value>Baby Girl</field-value>
    </value>
  </field>
  <field sbr="true" matching="true" blocking="true">
    <name>Person.SSN</name>
    <value>
      <file delimiter="|">
        <file-name>SSN.txt</file-name>
      </file>
    </value>
</exclusion-List>
```

The filter.xml file provides a simple format for you to define filters for survivor calculation, blocking queries, and matching. The following sample defines a rule that filters out the listed values from the FirstName field in the SBR. It defines a second rule that filters out the values listed in FirstNameFilters.txt from the SSN field.

The filter values file for the SSN field in the above example would look similar to the following:

```
000000000|111111111|222222222|333333333|444444444|555555555|666666666|777777777|
888888888|999999999
```

Filter Definition Elements

The following table lists and describes the XML elements and attributes that define the filters to be used for the SBR, blocking query, or match process. You can either define the exclusion lists directly in filter.xml or in a flat file that is referenced from 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 the *Oracle Healthcare Master Person Index Configuration Reference*.
value	A list of field-value elements that specify the values to filter.

Element Description	Attribute
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.

Configuring Master Person Index Update Policies

When update.xml is generated, no Java classes are defined for the update policies. You can create custom update policy classes and specify that the custom classes be used instead. Custom update policies must implement com.sun.mdm.index.update.UpdatePolicy and must be defined as custom classes in the Oracle Healthcare Master Person Index project to be recognized as an update policy. The names of the custom plug-ins you create are the values you enter for the update policies. You can also set the update policy flag to specify whether the policies are performed when no changes are made to an existing record.

Perform the following tasks to configure the update policies:

- Defining Master Person Index Update Policies on page 8-3
- Setting the Master Person Index Update Policy Flag on page 8-4

Defining Master Person Index Update Policies

You can define update policies for any of the seven update policy elements. You do not need to specify a policy for each element.

To Define Update Policies

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **update.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **UpdateManagerConfig section** of the file.

3. Do any of the following:

   - To modify the merge policy for enterprise objects, change the value of the EnterpriseMergePolicy element to the fully qualified name of the new Java class.

     For example:

     ```
     <EnterpriseMergePolicy>com.sun.mdm.index.user.MyEntMergePolicy
        </EnterpriseMergePolicy>
     ```

   - To modify the unmerge policy for enterprise objects, change the value of the EnterpriseUnmergePolicy element to the fully qualified name of the new Java class.

     For example:

     ```
     <EnterpriseUnmergePolicy>com.sun.mdm.index.user.MyEntUnmergePolicy
     ```

```
   </EnterpriseUnmergePolicy>
```

- To modify the update policy for enterprise objects, change the value of the EnterpriseUpdatePolicy element to the fully qualified name of the new Java class.

  For example:

  ```
  <EnterpriseUpdatePolicy>com.sun.mdm.index.user.MyEntUpdatePolicy
     </EnterpriseUpdatePolicy>
  ```

- To modify the create policy for enterprise objects, change the value of the EnterpriseCreatePolicy element to the fully qualified name of the new Java class.

  For example:

  ```
  <EnterpriseCreatePolicy>com.sun.mdm.index.user.MyCreatePolicy
     </EnterpriseCreatePolicy>
  ```

- To modify the merge policy for system objects, change the value of the SystemMergePolicy element to the fully qualified name of the new merge policy Java class.

  For example:

  ```
  <SystemMergePolicy>com.sun.mdm.index.user.MySysMergePolicy
     </SystemMergePolicy>
  ```

- To modify the unmerge policy for system objects, change the value of the SystemUnmergePolicy element to the fully qualified name of the new Java class.

  For example:

  ```
  <SystemUnmergePolicy>com.sun.mdm.index.user.MySysUnmergePolicy
     </SystemUnmergePolicy>
  ```

- To modify the assumed match policy, change the value of the UndoAssumeMatchPolicy element to the fully qualified name of the new Java class.

  For example:

  ```
  <UndoAssumeMatchPolicy>com.sun.mdm.index.user.MyUndoAsmMatchPolicy
     </UndoAssumeMatchPolicy>
  ```

4. Save and close the file.

Setting the Master Person Index Update Policy Flag

The update flag determines whether update policies are performed against a record when a transaction does not cause any changes to the record's data.

To Set the Update Policy Flag

1. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **update.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **UpdateManagerConfig section** of the file.

3. To specify that update policies are not performed when no updates are made, set the SkipUpdateIfNoChange element to **true**.

   For example:

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

4. To specify that update policies are performed even though no updates are made, set the SkipUpdateIfNoChange element to **false**.

   For example:

   ```
   <SkipUpdateIfNoChange>false</SkipUpdateIfNoChange>
   ```

5. Save and close the file.

Defining Custom Field Validations for the Master Person Index

By default, validation.xml defines one validation rule named "validate-local-id". This rule defines certain validations to perform 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.

Note: On most of the screens, user input is validated using the following regular expression:

```
^[a-zA-Z0-9.,()\\-\\/+=:/\@&_ %~'\r\n\\[\\]\\p{L}]*$
```

You can define additional validations for your data by creating the Java class and appropriate methods. Create the custom class in the Source Package folder of the master person index EJB project so you can include the custom validation rule in the master person index application code package. 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.

To Implement a Validation Rule

1. Create the custom validation rules in the Source Package folder of the EJB project.
2. In the Projects window, expand the **Configuration node** in the project you want to modify, and then double-click **validation.xml**.

   The file opens in the NetBeans XML editor.
3. Create a new rule element in the rules element.
4. In the new rule element, create the following attributes:
   - **name** - A unique name for the validation rule.
   - **object-name** - The name of the master person index Java class that defines the object to which the validation rule is applied, such as SystemObject or Parent_NameObject (where Parent_Name is the name of the parent object in the object definition)
   - **class** - The complete path of the Java class to call for the validation rule.

     For example:

     ```
     <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>
```

5. Save and close the file.

9

Master Index Data Manager Configuration

This chapter provides the information you need to configure the Master Index Data Manager (MIDM).

This chapter includes the following sections:

Note: For search screen configuration, if any of the following elements are configured to true, description value of <field-group> should not be null.

- show-euid
- show-lid
- show-status
- show-create-date
- show-create-time

Configuring the Master Index Data Manager Appearance

You can modify the configuration of the fields and objects to specify how they appear on the MIDM pages by modifying `midm.xml`. You can perform any of the following actions to customize the general appearance of the MIDM. Though some appearance options can be configured using the Configuration Editor, it is best to modify the XML file directly.

Perform any of the following tasks to configure the appearance of the MIDM.

Configuring Locale for MIDM User

While configuring data model or project, you can set the default language for MIDM. To do so, configure the value for <default-locale> in the midm.xml file. If you do not configure the locale, English (en) will be used as the default language. The following languages are supported:

- English (en)
- German (de)
- French (fr)
- Japanese (ja)
- Chinese (zh)
- Spanish (es)
- Portuguese (pt)

Adding Objects to the MIDM

You can define additional objects for the MIDM as long as those objects are defined in object.xml. Each object can only contain the fields that are also defined for that object in `object.xml`. If you add objects to the object structure using the Configuration Editor, the new objects are automatically added to the `midm.xml` and you do not need to perform these steps.

To Add an Object to the MIDM

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. In the nodes list of the file, create a new node element.
3. In the new node element, create and define the following elements.
   - **name** - The name of the object.
   - **display-order** - The order in which the child object types are displayed on the MIDM pages.

For example:

```
<node>
    <name>Phone</name>
    <display-order>3</display-order>
</node>
```

Note: You might need to renumber any existing objects to keep the numbering sequential.

4. Define fields for the new object, as described in Adding Fields to the MIDM on page 9-5.
5. Define the relationship of the object, as described in Defining MIDM Object Relationships on page 9-14.
6. Save and close the file.

Modifying MIDM Objects

Once an object is defined in `midm.xml`, you can modify the name or display order. Only modify the name of an object if you modify the corresponding object name in `object.xml` and the remaining configuration files. Do not change the parent object name. If you modify an object in the object definition on the Configuration Editor, the `midm.xml` is automatically updated as well.

To Modify an MIDM Object

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.
2. To modify the object name, change the value of the name element to the new name for the object.

   Note: If you change an object name, modify the name in the relationships section as well.
3. To modify the order in which the object appears on the MIDM, change the value of the display-order element.
4. If necessary, renumber the order of other child objects so they are sequential.
5. Save and close the file.

Adding Object and Sub-object Display Names for the MIDM

Once an object or sub-object is defined in `midm.xml`, you can add a Display Name for how it will appear in the MIDM. There are two ways to do this.

To Add Object and Sub-object Display Names for the MIDM Using the Configuration Editor

1. In the Projects window, right-click the Configuration node in the project you want to modify, and then select **Edit**.

2. To add a Display Name, select the primary object or a sub-object in the Object Definition pane.
3. In the Properties pane change the value of the Display Name element to the name you want for the object or sub-object.

   The name you set here is the name that will appear as the Display Name for the object or sub-object in the MIDM.
4. Click **Save**.

To Add Object and Sub-object Display Names for the MIDM Using the midm.xml File

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.
2. To add a Display Name, change the value of the name element to the name for the object or sub-object.

   The name you set here is the name that will appear as the display name for the object or sub-object in the MIDM.
3. Save and close the file.

Deleting Objects From the MIDM

Once an object is defined in `midm.xml`, you can remove the object. If the object remains defined in `object.xml`, then the object is still a part of the enterprise record, but does not appear on the MIDM. Before removing an object from the `midm.xml`, make sure none of its fields are required in order to create a new record. If you delete an object using the Configuration Editor, the object is automatically deleted from `midm.xml`.

To Delete an Object

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.
2. Scroll to the node element naming the object to delete.
3. Delete all text between and including the node element for that object.

   Using the sample below, to delete the Phone object, delete all the text in the sample.

```
<node>
   <name>Phone</name>
   <display-order>3</display-order>
   <field>
      <name>PhoneType</name>
      <display-name>Phone Type</display-name>
      <display-order>1</display-order>
      <max-length>8</max-length>
      <gui-type>MenuList</gui-type>
      <value-list>PHONTYPE</value-list>
      <value-type>string</value-type>
      <key-type>true</key-type>
   </field>
   <field>
      <name>Phone</name>
```

```
      <display-name>PhoneNumber</display-name>
      <display-order>2</display-order>
      <max-length>20</max-length>
      <gui-type>TextBox</gui-type>
      <value-type>string</value-type>
      <input-mask>(DDD)DDD-DDDD</input-mask>
      <value-mask>xDDDxDDDxDDDD</value-mask>
   </field>
</node>
```

4. If necessary, renumber the order of the remaining objects so they are sequential.
5. Remove the object from the relationship definition, as described in Defining MIDM Object Relationships and Defining Relationships Between Master Person Index Objects.
6. Save and close the file.

Adding Fields to the MIDM

You can define new fields for an object in `midm.xml`, but the field must correspond with a field defined for that object in `object.xml`. Only the fields defined in `midm.xml` appear on the MIDM windows. If you add a field to the object structure using the Configuration Editor, it is automatically added to `midm.xml`.

To Define New Fields

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.
2. Scroll to the node element that defines the object to which you want to add a field.
3. In the node element, create a new element named field.
4. Define and configure the elements listed in MIDM Field Configuration Elements for the new field-field_name element.

   For example:

```
<field
   <name>LastName</name>
   <display-name>LastName</display-name>
   <display-order>2</display-order>
   <max-length>40</max-length>
   <gui-type>TextBox</gui-type>
   <value-type>string</value-type>
   <is-sensitive>false</is-sensitive>
   <is-global-sensitive>false</is-global-sensitive>
   <key-type>false</key-type>
</field>
```

5. If necessary, renumber any existing fields to keep the numbering sequential.
6. Save and close the file.

MIDM Field Configuration Elements

The following table lists and describes the XML elements that define the fields that appear on the Master Index Data Manager.

Element	Description
name	The name of the field as it appears in the object definition.
display-name	The name of the field as it will appear on the MIDM.
display-order	The order in which the field appears in the object 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. The display order goes from left to right,
max-length	The maximum number of characters displayed on the MIDM for the field.
gui-type	An indicator of the type of display for the field. Specify **TextBox** for a standard data entry field; **MenuList** for a field that must be populated by selecting from a drop-down list; or **TextArea** for a long field that requires a scrollbar, such as a comments field.
value-type	The Oracle Healthcare Master Person Index data type for the data populated in the field. Enter one of the following values: ■ **string** - Field contains a string of characters. ■ **date** - Field contains a date value. ■ **float** - Field contains a floating point integer. ■ **int** - Field contains an integer. ■ **char** - Field contains a single character. ■ **boolean** - Field contains either "true" or "false".
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 no punctuation is entered by the user. To define an input mask, type a character type for each character in the field, and place any necessary punctuation between the character types. For example, the input mask for the above telephone format is "(DDD)DDD-DDDD". The following character types are allowed: ■ **D** - indicates a numeric character. ■ **L** - indicates an alphabetic character. ■ **A** - indicates an alphanumeric character. **Note**: If the length of the input mask is greater than the value specified for the max-length element, update the value of the max-length element 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. This mask ensures that data is stored in the database in the correct format. This mask must be the same length as the input mask. To specify a value mask, type the same value as is entered for the input mask, but type an "x" in place of each punctuation mark. For example, if an SSN field has an input mask of DDD-DD-DDDD, you need to specify a value mask of DDDxDDxDDDD to strip the dashes before storing the SSN. A value mask is not required for date fields.
value-list	The name of the menu list used to populate the drop-down list for the field. This is required if the gui-type specified for the field is MenuList, and it must match a code of an element in the sbyn_common_header database table.

Element	Description
is-sensitive	An indicator of whether the value of the field is hidden on the MIDM for users without "View_Sensitive" permission. is-sensitive works in conjunction with the condition specified in the `<sensitive-mask-condition>` attribute, which differentiates is-sensitive from is-global-sensitive. Users with this security permission 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, and is only applicable for is-sensitive for backward compatibility.
is-global-sensitive	An indicator of whether the value of the field is hidden on the MIDM for users without "View_Global_Sensitive" permission. Users with this security permission can view the hidden information. Specify **true** to hide the field value; specify **false** (or remove the is-global-sensitive element) to display the field value.
display-length	You have the capability to limit the number of characters that will display for each field in the MIDM.
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. For example, if the object structure contains a child object for addresses and each record can have only one of each type of address (for example, one home address and one work address), then the address type field would be a key field. Specify **true** to indicate the field is a key field; specify **false** if it is not.

Removing Fields From the MIDM

If a field is defined for an object in `midm.xml`, that field appears on the MIDM windows. If there are any fields defined in the object structure that you do not want to display on the MIDM, you can remove the field definition from `midm.xml`. If you remove a field from the object structure using the Configuration Editor, it is automatically removed from `midm.xml`.

> **Note:** If you remove a unique key type field, you must write a custom plug-in that will automatically populate a value into the field in order to meet unique key type field constraints.

To Remove a Field From the MIDM

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the field element that defines the field you want to remove.
3. Delete the field elements that define the field you want to remove.
4. If necessary, renumber the remaining fields, as described in Modifying MIDM Field Display Options on page 9-8.
5. Save and close the file.

Modifying MIDM Field Display Options

Once a field is defined for an object in `midm.xml`, you can change the name that appears on the MIDM for that field, the location of the field, and the type for the field (such as text box, menu list, and so on).

To Modify a Field's Display Options

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the node element that defines the object that contains the field you want to modify.
3. Scroll to the field element you want to modify.
4. To modify the field label, change the value of the **display-name element**.

   For example:

   ```
   <display-name>Last Name</display-name>
   ```

5. To modify the location of the field on the MIDM, change the value of the **display-order element**.

   For example:

   ```
   <display-order>1</display-order>
   ```

   Note: If you change the order of one field, you must change the order of at least one other field to maintain sequential numbering. For example, if you change a field's location from 2 to 1, you must then change the location of the field originally specified for location 1.

6. To modify the type of field to display, change the value of the **gui-type element**.

   For example:

   ```
   <gui-type>TextBox</gui-type>
   ```

   Note: You can enter `TextBox` (for a standard field), `MenuList` (for a field whose value must be selected from a drop-down list), or `TextArea` (for a long field that requires a scrollbar).

7. Save and close the file.

Setting Child Object Fields to Display in the MIDM

By default, the Required property for a child object is **false**. If you wish the child object to always have data or be required, set the Required property to **true**. The property below the Required property is Used In MIDM. The default for this property is **true**. However, you may not want certain child object information displayed on the MIDM, and in some cases, such as blobs, the information does not display well in the MIDM. Like the Required property, you have the option to change the value for Used In MIDM. You can perform these tasks in the OHMPI new project wizard when you

create a project, and then you can change these settings in the Configuration Editor after the project has been created.

To Set a Child Object to Display in the MIDM Using the OHMPI New Project Wizard

When you create a project from scratch in the OHMPI new project wizard, if you want to display the child object's values in the MIDM, do the following:

- During the creation of a project, and with a child object selected in the Object Definition column, change the Required property's value to **true**.

As mentioned above, the Used In MIDM property for the selected child object has a default of **true**, meaning that its values will appear in the MIDM. If you do not want the values to be displayed in the MIDM, do the following:

- During the creation of a project, and with a child object selected in the Object Definition column, change the Used In MIDM property's value to **false**.

> **Note:** If the Used In MIDM value is set to **false**, the child object is removed from the Relationship section of the `midm.xml` file.

To Set a Child Object to Display in the MIDM Using the Configuration Editor

After you have created and deployed a project using the OHMPI new project wizard, you have the capability of changing the values you set for the child object's Required and Used In MIDM properties by doing the following in the Configuration Editor:

- With a child object selected in the Object Definition column, change the Required property's value to **true** if it was previously set to **false** and you now want the child object's values to display in the MIDM.

As mentioned above, the Used In MIDM property for the selected child object has a default of **true**, meaning that its values will appear in the MIDM. If you do not want the values to be displayed in the MIDM, after a project has been created, do the following in the Configuration Editor:

- With a child object selected in the Object Definition column, change the Used In MIDM property's value to **false**. Conversely, if you want the child object values to display in the MIDM, set the value to **true**

> **Note:** If the Used In MIDM value is set to **false**, the child object is removed from the Relationship section of the `midm.xml` file.

Specifying a Drop-Down List for an MIDM Field

Once a field is defined for an object in `midm.xml`, you can specify or change the name of the drop-down list for the field. If you modify the Code Module for the field in the Configuration Editor, the drop-down list is automatically updated in `midm.xml`.

To Specify a Drop-Down List

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the node element that defines the object containing the field you want to modify.

3. Scroll to the field element you want to modify.
4. Enter the name of the drop-down list in the value-list element.

   If the element does not exist for the field, create a new value-list element. For example:

   ```
   <value-list>STATE</value-list>
   ```

   Note: The value of the gui-type element for the field must be "MenuList" if you specify a drop-down list. The value-list element must match a code column value in the sbyn_common_header database table unless the drop-down list is populated by information in the sbyn_user_code table (as they might be for auxiliary IDs). In this case, the value-list element must match a code_list column value in sbyn_user_code.

5. Save and close the file.

Specifying an MIDM Field's Length and Format

Once a field is defined for an object in `midm.xml`, you can change the number of characters that can be entered for the field in the MIDM. You can also specify whether to automatically enter punctuation into a field on the MIDM, but remove the punctuation in the database. If you modify the field's length and format in the object structure using the Configuration Editor, `midm.xml` is automatically updated as well.

Note: Field length here is constrained by the length of the database column containing the field and the length defined in `object.xml`. This applies to the max-length element and the input and value masks.

To Modify a Field's Length and Format

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.
2. Scroll to the node element that defines the object containing the field you want to modify.
3. Scroll to the field element you want to modify.
4. To modify the length of a field, change the value of the max-length element.

   For example:

   ```
   <max-length>100</max-length>
   ```

5. To modify the format of the field, change the value of the input-mask and value-mask elements.

   If these elements do not exist for the field, create new input-mask and value-mask elements. For example:

   ```
   <input-mask>(DDD)DDD-DDDD</input-mask>
   <value-mask>xDDDxDDDxDDDD</value-mask>
   ```

Note: If an input mask is defined, in most cases a value mask must also be defined. If the input mask exceeds the length specified by the max-length element, update the max-length element to match. For information about input and value masks, see MIDM Field Configuration Elements on page 9-5.

6. Save and close the file.

Modifying an MIDM Field's Data Type

Each field on the MIDM requires a specific type of data to be entered. For example, name fields generally require a data string and date fields require a valid date or numeric characters. The type of data defined for each field must correspond with the field type defined for that field in `object.xml` and in the database. If you modify a field's data type in the object structure using the Configuration Editor, `midm.xml` is automatically updated as well.

To Modify the Data Type

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the node element that defines the object that contains the field you want to modify.
3. Scroll to the field element you want to modify.
4. Change the value of the value-type element.

   For example:

   ```
   <value-type>string</value-type>
   ```

5. Save and close the file.

Defining Key Fields for an Object

You can specify that a certain field or combination of fields be unique in a system object or SBR. An example of a unique fields would be the address type if only one address of each type is allowed. A field's key type status in `midm.xml` must match its key type status in `object.xml`. If you modify a field's key type status in the object structure using the Configuration Editor, `midm.xml` is automatically updated as well.

To Modify the Key Status

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the node element that defines the object containing the field you want to modify.
3. Scroll to the field element you want to modify.
4. To specify that a field must be unique in a system object, change the value of the key-type element to **true**.

5. To specify that a field does not need to be unique in a system object, change the value of the key-type element to **false**.

   For example:

   ```
   <key-type>false</key-type>
   ```

6. To specify that a combination of fields must be unique for an object rather than just one single field, set the key-type element to **true** for each field.
7. Save and close the file.

Masking Field Values on the MIDM

You can specify that the values of certain fields be hidden on the MIDM from users without the appropriate access permissions. You can create a custom plug-in that uses the value of the VIP Flag field to determine whether the values of specified fields are hidden (typically in records with a VIP status of "VIP" or "Employee"). In records with any other VIP status, the field values would be visible regardless of whether the field is marked for masking.

This option is only available if the object-sensitive-plug-in-class element in the impl-details section is populated with the custom plug-in class (by default, this element is empty).

> **Note:** To mask field values, you need to define a custom plug-in to implement the masking rules.

To Mask Field Values on the MIDM

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.
2. Scroll to the node element that defines the object containing the field you want to modify.
3. Scroll to the field element you want to modify.
4. Change the value of the is-sensitive element to **true**.

   If the element does not exist for the field, create a new is-sensitive element. For example:

   ```
   <is-sensitive>true</is-sensitive>
   ```

   > **Note:** The is-sensitive element must appear before the key-type element in the field definition.

5. Save and close the file.

Setting Conditional Masking on the MIDM

The `<impl-details>` section of the `midm.xml` file is where you set up the field called VIPFlag for conditional masking for each record in the MIDM. Use the VIPFlag (see the example below) to control the masking of fields that have `<is-sensitive>` set to **true**. For example:

- If a record's field such as SSN has is-sensitive set to **true**, and if the VIPFlag value is set to **false** in the `midm.xml` file, the SSN (123-12-1234) is not masked in the MIDM.
- If a record's SSN has is-sensitive set to **true**, and if the VIPFlag value is set to **true** in the `midm.xml` file, the SSN (xxx-xx-xxxx) is masked in the MIDM.

> **Note:** The field **VIPFlag** is a placeholder, and you can change this field name to whatever best suits your configuration requirements (for example, to **MVP**). The same is true with the **true** or **false** values. They can also be changed (for example, to **yes** or **no**).

```
<impl-details>
    <master-controller-jndi-name>ejb/PatientMasterController</master-controller
-jndi-name>
    <validation-service-jndi-name>ejb/PatientCodeLookup</validation-service
-jndi-name>
     <usercode-jndi-name>ejb/PatientUserCodeLookup</usercode-jndi-name>
    <reportgenerator-jndi-name>ejb/PatientReportGenerator</reportgenerator
-jndi-name>
<object-sensitive-plug-in-class></object-sensitive-plug-in-class>
    <sensitive-mask-condition>
        <sensitive-field>
            <name>VIPFlag</name>
            <value>true</value>
    <sensitive-mask-condition>
</impl-details>
```

> **Note:** `<sensitive-mask-condition>` is used with the `<is-sensitive>` attribute to provide conditional masking for sensitive fields. For example, if you wish to mask the SSN of certain patients, you can add `<is-sensitive>` to the `<SSN field>` and provide a mask condition. In the `<sensitive-mask-condition>` example illustrated above, we are masking the SSN for all patient records whose **VIPFlag** is set to **true** or **Yes**.

Configuring Search Result Pages

The sensitive field added to the `<sensitive-mask-condition>` tag needs to be added to the `<search-result-pages>`. In the example below, the Person.VIPFlag sensitive field is added to the `<search-result-pages>` tag.

```
<search-result-pages>
        <search-result-list-page>
                <search-result-id>0</search-result-id>
                <item-per-page>10</item-per-page>
                <max-result-size>100</max-result-size>
                <field-group>
                        <description></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-ref>Person.VIPFlag</field-ref> <-- sensitive mask
field needs to be added to the search result pages.
                </field-group>
```

```
        </search-result-list-page>
</search-result-pages>
```

Defining MIDM Object Relationships

The relationships in `midm.xml` are predefined based on the information you provided when you created the object structure definition (`object.xml`). The relationship structure in `midm.xml` should match that of `object.xml`.

To Define Relationships

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the relationships element.

3. Specify the name of the parent object in the name element.

4. Specify the name of the child objects in the children elements.

   For example:

   ```
   <relationships>
      <name>Person</name>
      <children>Alias</children>
      <children>Address</children>
      <children>Phone</children>
      <children>AuxId</children>
   </relationships>
   ```

5. To remove a child object from the relationships list, delete the children element defining the object you want to delete.

6. Save and close the file.

Defining Relationships Between Master Person Index Objects

Once all objects are customized, you must define relationships between those objects if you are modifying the XML file directly. If you are using the Configuration Editor, the relationships are automatically defined in the object tree.

To Define Object Relationships (XML Editor)

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **object.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **relationships element** near the end of the file.

3. To specify a new parent object, modify the value of the name element.

   For example:

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

Note: This is not recommended. Changing the parent name requires changing all instances of the name in all configuration files. To change the parent object name, use the Configuration Editor, which automatically propagates the change.

4. To change the name of a child object, modify the value of the appropriate children element.
5. To add a child object, create and name a new children element.
6. To delete a child object, delete all text between and including the appropriate children element.
7. Save and close the file.

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

Defining MIDM Local ID Labels

The local-id element, which is nested in the gui-definition element, defines alternate names for the local ID headings and labels. The name defined here replaces the default local ID heading and field names on all pages, including the search result column names. This element is optional, and if it does not exist the label names default to Local ID.

To Define Local ID Labels

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.
2. Scroll to the **local-id element** in the gui-definition element.
3. Enter the name you want to appear on the MIDM for the local ID fields and headings.
4. Save and close the file.

Configuring the Master Index Data Manager Pages

You can configure certain pages that appear on the MIDM by modifying the `midm.xml`. Each page on the MIDM is configured separately, though many contain the same XML elements and attributes. For the Record Details page, you can configure and create search pages and search results lists. If you add a new query to `query.xml` and you want to access that query from the MIDM, you need to create a new search page for the query on the Record Details page. The Dashboard page cannot be configured.

Perform any of the following actions to configure the pages of the MIDM.

- Specifying the Initial View for the MIDM on page 16
- Configuring the MIDM Duplicate Records Page on page 16
- Configuring the MIDM Record Details Page on page 20
- Creating Search Pages on the Record Details Page on page 20

Specifying the Initial View for the MIDM

By default, the Record Details page appears when a user logs in to the Master Index Data Manager. You can specify any of the other tabbed pages as the initial view. Each page on the MIDM is identified by a screen ID in `midm.xml`, which is an integer defined in the screen-id element for each page. Use this ID to specify which page to display first when a user logs in.

To Specify the Initial View

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Locate the page definition for the page you want to display first, and note the value of its screen-id element.
3. Scroll to the **initial-screen-id element**, located in the gui-definition element.
4. Enter the value of the screen-id element for the page you want to display first.
5. Save and close the file.

Configuring the MIDM Duplicate Records Page

The Duplicate Records page allows you to search for potential duplicate records, view a results list, and then view a comparison of the records you select. It also provides features to resolve or merge the potential duplicates in question. You can configure several aspects of the Duplicate Records page, including display options, search pages, and the results list.

The following topics provide instructions for each type of configuration:

Configuring Duplicate Records Display Options

You can configure certain display options for the Duplicate Records page, such as the name of the tabbed heading and the order in which the tab appears on the MIDM. Below is a sample of the display option elements for the Duplicate Records page.

```
<duplicate-records>
   <root-object>Person</root-object>
   <tab-name>Duplicate Records</tab-name>
   <screen-id>3</screen-id>
   <display-order>1</display-order>
   ...
```

```
</duplicate-records>
```

You should never need to modify the root-object or screen-id element for the Duplicate Records page.

To Configure Duplicate Records Display Options

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **duplicate-records element** (located in the gui-definition element).

3. To specify a different name for the Duplicate Record tab, modify the value of the tab-name element.

4. To change the order in which the Duplicate Records tab appears on the MIDM, change the value of the display-order element.

   Note: You might need to renumber the remaining tabs to keep the numbering sequential.

5. Save and close the file.

Configuring Duplicate Records Search Pages

For the Duplicate Records search pages, you can configure the names of the searches that can be performed, the fields to use as search criteria (within a limited set of system fields), and instructions for performing a search. Below is a sample of the search page configuration for the Duplicate Records page.

```
<duplicate-records>
   ...
      <simple-search-page>
         <screen-title>Advanced Search</screen-title>
         <search-result-id>0</search-result-id>
         <search-screen-order>0</search-screen-order>
         <show-euid>true</show-euid>
         <show-lid>true</show-lid>
         <show-status>true</show-status>
         <show-create-date>true</show-create-date>
         <show-create-time>true</show-create-time>
         <instruction></instruction>
         <field-group/>
      </simple-search-page>
      ...
</duplicate-records>
```

To Configure the Duplicate Records Search Page

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **duplicate-records element** (located in the gui-definition element).

3. Modify any of the elements described in Duplicate Records Search Page Elements.

4. Save and close the file.

Configuring the Duplicate Records Results List

For the Duplicate Records results list, you can configure the number of results to display at one time, the total number of results a search can return, and the fields that appear in the results list. Below is a sample results list configuration for the Duplicate Records page.

```
<duplicate-records>
   ...
   <search-result-pages>
      <search-result-list-page>
         <search-result-id>0</search-result-id>
         <item-per-page>10</item-per-page>
         <max-result-size>100</max-result-size>
         <field-group>
            <description></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>
</duplicate-records>
```

To Configure the Duplicate Records Results List

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **duplicate-records element** (located in the gui-definition element).
3. Modify any of the elements described in Duplicate Records Search Results List Elements.
4. To add a field to the results list, create and name a new field-ref element in the field-group element.
5. To delete a field from the results list, delete the field-ref element defining the field.
6. Save and close the file.

Duplicate Records Page Configuration Elements

The following two tables describe the configuration elements in `midm.xml` for the Duplicate Records search and results pages. The same elements are also used for the Source Record View/Edit page.

Duplicate Records Search Page Elements

The following table lists and describes the configurable elements for the Duplicate Records and Source Record search pages.

Table 8-1 Duplicate Records Search Page Configuration

Element/Attribute	Description
search-pages	A container element that lists and defines the searches that are available from the page.

Element/Attribute	Description
simple-search-page	A container element that defines one type of search for the Duplicate Records page. You can define multiple simple search pages.
screen-title	The name of the search as it appears on the search page. MIDM users can select a type of search to perform based on the titles you define for the searches here.
search-screen-order	The order in which the search page appears on the MIDM.
show-euid	An indicator of whether to display the EUID field as search criteria. Specify **true** to display the EUID; otherwise specify **false**.
show-lid	An indicator of whether to display the local ID and system fields as search criteria. Specify **true** to display the fields; otherwise specify **false**.
show-status	An indicator of whether to display the record status field as search criteria. Specify **true** to display the field; otherwise specify **false**.
show-create-date	An indicator of whether to display the create date field to allow searching on the date the potential duplicate flag was created. Specify **true** to display the field; otherwise specify **false**.
show-create-time	An indicator of whether to display the create time field to allow searching on the time the potential duplicate flag was created. Specify **true** to display the field; otherwise specify **false**.
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 can be contained in a labeled box on the Search 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.

Duplicate Records Search Results List Elements

The following table lists and describes the configurable elements for the Duplicate Records search results list.

Table 8-2 Duplicate Records Search Results Configuration

Element/Attribute	Description
search-result-pages	A container element for a list of search results page definitions.
search-result-list-page	A container element for the configuration information for the search results page.
item-per-page	The number of resulting records to display on one page.
max-result-size	The maximum number of records to return for a search.
field-group	A container element for a list of fields that appear in the search results.
description	A brief description for the field group. This value appears on the MIDM above the fields that are returned from a search.

Element/Attribute	Description
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.

Configuring the MIDM Record Details Page

You can configure certain display options for the Record Details page, such as the name of the tabbed heading and the order in which the tab appears on the MIDM. You can also create, modify, and delete search pages and search results lists. Search configuration is described in Creating Search Pages on the Record Details Page and Modifying a Search Page on the Record Details Page.

Below is a sample of the display option elements for the Record Details page.

```
<record-details>
   <root-object>Person</root-object>
   <tab-name>Record Details</tab-name>
   <screen-id>1</screen-id>
   <display-order>2</display-order>
   ...
</duplicate-records>
```

You should never need to modify the root-object or screen-id element.

To Configure Record Details Display Options

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **record-details element** (located in the gui-definition element).
3. To specify a different name for the Record Details tab, modify the value of the tab-name element.
4. To change the order in which the Record Details tab appears on the MIDM, change the value of the display-order element.

   Note: You might need to renumber the remaining tabs to keep the numbering sequential.

5. Save and close the file.

Creating Search Pages on the Record Details Page

Several search pages are created by the Master Person Index Wizard based on the information you specify. You can create and customize new search pages for the Record Details page. Each search you define must use a query that is defined in `query.xml`.

Follow these steps to create a new search page:

- Step 1: Define the Search Page on page 21
- Step 2: Define the Search Fields on page 21
- Step 3: Specify Search Options on page 22

Step 1: Define the Search Page

The first step in creating a search page is to define certain properties for the appearance of the page, such as its name, whether to display the EUID or local ID field, and general instructions for the search.

> **Note:** If either the EUID field or the local ID and system fields appear on a search page, any values entered into these fields take precedence over information entered into other search fields. For example, if an invalid EUID is entered but valid first and last names are entered, no results are returned due to the invalid EUID. The EUID field takes precedence over the local ID and system fields.

To Define the Search Page

1. In the Projects window, expand the Configuration node in the Project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **record-details element**, and then create a **simple-search-page element**.

   Make sure the new element falls within the record-details element, but outside any existing simple-search-page elements. For example:

```
<record-details>
   <simple-search-page>
    ...
   </simple-search-page>
   <simple-search-page>
   </simple-search-page>
</record-details>
```

3. In the new simple-search-page element, create the elements listed in Record Details Search Page Definition Elements on page 9-23 and enter the appropriate value for each element.

   For example:

```
<record-details>
   <simple-search-page>
    ...
   </simple-search-page>
   <simple-search-page>
      <screen-title>Address Search</screen-title>
      <search-result-id>1</search-result-id>
      <show-euid>true</show-euid>
      <show-lid>false</show-lid>
      <instruction>Enter address information below.</instruction>
   </simple-search-page>
</record-details>
```

4. Continue to Step 2: Define the Search Fields on page 9-21.

Step 2: Define the Search Fields

Once you define the search page, you need to specify the fields that appear on the page. Fields are specified in field groups, and each field group represents an area on the search page. All fields specified for a field group appear in the area defined by that group. The label for the area is defined by the description of the field group.

To Define Search Fields

1. Complete Step 1: Define the Search Page.
2. In the new simple-search-page element, create a field-group element.

   For example:

```
<simple-search-page>
   <screen-title>Simple Person Search</screen-title>
   <search-result-id>2</search-result-id>
   <search-screen-order>1</search-screen-order>
   <show-euid>false</show-euid>
   <show-lid>false</show-lid>
   <field-group>
   </field-group>
</simple-search-page>
```

3. In the new field-group element, create the elements and attributes listed in Record Details Search Field Definition Elements on page 9-24 and enter the appropriate value for each.

   For example:

```
<simple-search-page>
   ..
   <field-group>
      <description>Address</description>
      <field-ref>Address.AddressType</field-ref>
      <field-ref>Address.AddressLine1</field-ref>
      <field-ref>Address.AddressLine2</field-ref>
      <field-ref required="true">Address.City</field-ref>
      <field-ref>Address.State</field-ref>
   </field-group>
</simple-search-page>
```

4. Repeat steps 2 and 3 for each field group you want to display on the selected search page.
5. Continue to Step 3: Specify Search Options.

Step 3: Specify Search Options

After you define the criteria fields for the MIDM search, you can specify certain options for the search, such as the types of available searches, whether each search is weighted, and whether the search allows wildcard characters.

> **Note:** Wildcards should not be allowed for blocking queries or phonetic searches.

To Specify Search Options

1. Complete Step 2: Define the Search Fields.
2. In the simple-search-page element you created, create a search-option element.

   For example:

```
<simple-search-page>
   <screen-title>Simple Person Search</screen-title>
   <search-result-id>2</search-result-id>
   <search-screen-order>1</search-screen-order>
   <show-euid>false</show-euid>
```

```
   <show-lid>false</show-lid>
   <field-group>
    ...
   </field-group>
   <search-option>
   </search-option>
</simple-search-page>
```

3. In the new search-option element, create the elements listed in Record Details Search Option Elements on page 9-25 and enter the appropriate value for each element.

   For example:

```
<search-option>
   <display-name>Alpha Search</display-name>
   <query-builder>ALPHA-SEARCH</query-builder>
   <weighted>false</weighted>
   <parameter>
     <name>UseWildCard</name>
     <value>true</value>
   </parameter>
</search-option>
```

4. Repeat the previous two steps for each search type you want to make available on the selected search page.

   Note: If you define multiple search option elements, an option button (labeled by the value of the display-name element) appears on the search page for each search option.

5. Save and close the file.

Record Details Search Page Definition Elements

The following table lists and describes the elements you can configure in midm.xml to define the search pages on the Record Details page.

Element	Description
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.
search-screen-order	The order in which the search page appears on the MIDM.
show-euid	An indicator of whether to display the EUID. Specify **true** to display the EUID; otherwise specify **false**. Only display this field if you want it to take precedence over all other search criteria.
show-lid	An indicator of whether to display the local ID and system fields. Specify **true** to display the fields; otherwise specify **false**. Only display these fields if you want them to take precedence over all other search criteria (except the EUID field).

Element	Description
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.

Record Details Search Field Definition Elements

The following table lists and describes the elements you can configure in `midm.xml` to define the fields for each search on the Record Details page.

Element/Attribute	Description
description	A description of the fields defined for the field-group element. This value appears as a box label for the area of the page that contains the specified fields.
field-ref	One field definition for a field in the field group. Use the simple field names of the fields 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 (*) on the search page. ■ **false** - The corresponding field is not required to perform the search. If the required attribute is not defined, the default is false. ■ **one of** - The corresponding fields with this designation are a group of search criteria, at least one of which is required to perform the search. This designation specifies that at least one field in the group of fields with the "one of" designation is required. If a group of fields is designated as "one of", those fields are marked with a dagger (†) on the search page. Tip: If you make a field required for a search, it is a good idea to make it required when creating a record as well by specifying true for the required property for the field in `object.xml`. Otherwise, searches performed from the MIDM could result in no possible matches even though possible matches exist.
field-ref/choice	An indicator of whether the field allows you to search by a range of values rather than an exact value. Specify any of the following values: ■ **exact** - The search is performed on the exact value entered (wildcard characters 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. If you specify "range" for a field in a search that uses a blocking query, be sure to modify the query block in `query.xml` accordingly. Tip: You can specify the same field for both exact and range searching by adding it twice to the field list with different attribute values, giving the choice of performing an exact search or a range search from the MIDM. For more information about range searching, see the *Oracle Healthcare Master Person Index Configuration Reference*.

Record Details Search Option Elements

The following table lists and describes the elements you can configure in `midm.xml` to define the attributes for each search on the MIDM, such as which query to use, whether the search results are weighted, and so on.

Element	Description
display-name	A short phrase describing the type of search to perform, such as "Alphanumeric Search" or "Phonetic Search". This appears next to the option button on the search page when multiple search options are defined.
query-builder	The type of query to use when this type of search is selected. The value entered here must match a query-builder name in `query.xml`.
weighted	An indicator of whether the results of the search are assigned matching probability weights. Specify **true** to assign matching weights; specify **false** to return unweighted results.
candidate-threshold	The maximum number of records to return for a search. This value must be a positive number and can only be 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, specified by name and value elements (described below).
name	The name of the parameter. Currently, only UseWildCard is available.
value	The value of the parameter. For the UseWildCard parameter, this is an indicator of whether the parameter is enabled or disabled. Specify **true** to allow wildcard characters; specify **false** to perform exact-match searches.

Modifying a Search Page on the Record Details Page

Once a search page is defined for the Record Details page, it can be modified as needed. Be sure to review the search pages automatically generated by the wizard to see if any further configuration is required. You can perform any of the following actions to customize existing search page elements.

- Modifying a Search Page Definition on page 25
- Modifying Search Fields on page 26
- Modifying Record Details Search Page Options on page 26

Modifying a Search Page Definition

Once a search page is defined for the Record Details page, you can modify the search page definition. The search page definition includes properties like the name of the page, the order of the search, the results list to use for the search, and so on.

To Modify a Search Page Definition

1. In the Projects window, expand the Configuration node in the Project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the simple-search-page element you want to modify in the record-details element.

3. In the simple-search-page element, change the value of any of the elements listed in Record Details Search Page Definition Elements.

For example:

```
<simple-search-page>
   <screen-title>Customer Search</screen-title>
   <search-result-id>2</search-result-id>
   <search-screen-order>2</search-screen-order>
   <show-euid>true</show-euid>
   <show-lid>false</show-lid>
   <instruction>Enter the EUID below.</instruction>
</simple-search-page>
```

4. Save and close the file.

Modifying Search Fields

Once field groups and fields are specified for a Record Details search page, you can modify the properties of the group and of the fields contained in a group. For more information about the elements that contain the search field configuration, see Record Details Search Field Definition Elements on page 9-24.

To Modify Search Fields

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the simple-search-page element you want to modify in the record-details element.

3. In the field-group you want to modify, do any of the following:

   - To modify the name of the area in which the field group appears in the MIDM, change the value of the description element.
   - To add a new field to a field group, create and name a new field-ref element in the appropriate field-group element.
   - To modify the name of a field defined for a field group, change the value of the field-ref element.
   - To specify whether a field is required, add a required attribute and specify a value defined in Record Details Search Page Definition Elements.
   - To specify whether a field is used for range searching, add a choice attribute and specify a value defined in Record Details Search Field Definition Elements.
   - To delete a field from a field group, delete all text between and including the field-ref tags that define the field to be deleted.
   - To delete an entire field group, delete all text between and including the field-group tags that define the field group to be deleted.

4. Save and close the file.

Modifying Record Details Search Page Options

Once search options are defined for a Record Details search page, you can modify those options if needed. For more information about the elements that define search options, see Record Details Search Option Elements.

To Modify Search Page Options

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the simple-search-page element you want to modify in the record-details element.

3. In the search-option element, and do any of the following:

   - To modify the name of the search option button, change the value of the display-name element.
   - To modify the query type of the selected search, change the value of the query-builder element. The query you specify must match a query defined in `query.xml`.
   - To specify that a search return weighted results, change the value of the weighted element to **true**.
   - To specify that a search return unweighted results, change the value of the weighted element to **false**.
   - To specify that wildcard characters can be used in a search, change the UseWildCard parameter value element to **true**.
   - To specify that wildcard characters cannot be used in a search, change the UseWildCard parameter value element to **false**.

4. Save and close the file.

Configuring the MIDM Assumed Matches Page

The Assumed Matches page allows you to search for assumed match records, view a results list, and then view the details for the record you select. You can also undo an assumed match if you think it was made in error. You can configure the name of the tabbed heading for the page, the display order, the number of records to return and display for a search, and the fields to display in the result list. The fields on the search page and in the results list are automatically generated and cannot be modified.

Below is a sample Assumed Matches page definition.

```
<assumed-matches>
   <root-object>Person</root-object>
   <tab-name>Assumed Matches</tab-name>
   <screen-id>4</screen-id>
   <display-order>3</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>100</max-result-size>
         <field-group>
            <description></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-ref>Person.Address.AddressLine1</field-ref>
            <field-ref>Person.Address.AddressLine2</field-ref>
```

```
        </field-group>
      </search-result-list-page>
   </search-result-pages>
</assumed-matches>
```

To Configure the Assumed Matches Page

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the assumed-matches element (located in the gui-definition element).

3. Make any of the following configuration changes:

   - To specify a new name for the Assumed Matches tab, modify the value of the tab-name element.
   - To change the order in which the Assumed Matches tab appears on the MIDM, change the value of the display-order element.

     Note: If you change the order of one tab, make sure to change the order of the remaining tabs so no tabs have an identical display order value.

   - To specify the number of records to display on the Assumed Matches search results list, change the value of the item-per-page element.
   - To specify a maximum number of records to return for an Assumed Matches search, change the value of the max-result-size element.
   - To add a search description, enter the description text into the description element.
   - To modify the fields that appear in the results list in addition to the system fields, change the value of an existing field-ref element, create a new field-ref element, or delete an existing field-ref element.

     For example:

     ```
     <field-group>
                 <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-group>
     ```

4. Save and close the file.

Configuring the MIDM Transactions Page

The Transactions page allows you to search for transaction records, view a results list, and then view the changes for the record you select. You can configure the name of the tabbed heading for the page, the display order, and the number of records to return and to display in the results list. The fields on the search page and results list are automatically generated and cannot be modified with the exception of the EUID, system, and local ID.

Below is the default Transactions page definition.

```
<transactions>
   <root-object>Person</root-object>
   <tab-name>Transactions</tab-name>
   <screen-id>2</screen-id>
   <display-order>4</display-order>
   <search-pages>
      <simple-search-page>
         <show-euid>true</show-euid>
         <show-lid>true</show-lid>
      </simple-search-page>
   </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>100</max-result-size>
         <field-group/>
      </search-result-list-page>
   </search-result-pages>
</transactions>
```

To Configure the Transactions Page

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **transactions element** (located in the gui-definition element).

3. Make any of the following configuration changes:

   - To specify a new name for the Transactions tab, modify the value of the tab-name element.
   - To change the order in which the Transactions tab appears on the MIDM, change the value of the display-order element.

     Note: If you change the order of one tab, make sure to change the order of the remaining tabs so no tabs have an identical display order value.

   - To include the EUID as a Transactions search field, change the value of the show-euid element to **true**.
   - To exclude the EUID from the Transactions search fields, change the value of the show-euid element to **false**.
   - To include the system and local ID as Transactions search fields, change the value of the show-lid element to **true**.
   - To exclude the system and local ID from the Transactions search fields, change the value of the show-euid element to **false**.
   - To specify the number of records to display on the Transactions search results list, change the value of the item-per-page element.
   - To specify a maximum number of records to return for a Transactions search, change the value of the max-result-size element.

4. Save and close the file.

Configuring the MIDM Reports Page

Configuring the Reports page consists of configuring the page itself and configuring the individual reports. Production reports are each configured in the same way in their own sub-screen definition element. Activity reports are configured together in one sub-screen definition. The sub-screen definitions means there are a series of tabbed pages within the Reports page for each different type of report.

Perform any of the following tasks to configure the reports:

- Configuring the Reports Page Definition on page 30
- Configuring Production Reports on page 30
- Configuring Activity Reports on page 31

Configuring the Reports Page Definition

For the Reports page, you can configure the name of the tabbed heading for the page and the order in which the tab appears on the MIDM. The report configuration section defines the appearance of the Reports page, and is located within a set of reports tags near the end of the file. Following is a sample of the Reports page configuration elements.

```
<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/>
```

To Configure the Reports Page Definition

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **reports element** (located in the gui-definition element).
3. To specify a different name for the Reports tab, modify the value of the tab-name element.
4. To change the order in which the Reports tab appears on the MIDM, change the value of the display-order element.

   Note: You might need to renumber the remaining tabs to keep the numbering sequential.

5. When you have finished configuring the Reports page, save and close the file.

Configuring Production Reports

A sub-screen on the Reports page is defined for each of the production reports. Use these sections to configure each production report to display information as you want to view it. You can also specify which reports can be run from the MIDM. Following is an example of a report configuration stanza.

```
<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>5</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></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.Address.AddressLine1</field-ref>
            <field-ref>Person.Address.AddressLine2</field-ref>
            <field-ref>Person.Address.City</field-ref>
         </field-group>
      </search-result-list-page>
   </search-result-pages>
</subscreen>
```

To Configure Production Reports

Perform the following steps for each production report.

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **reports element** (located in the gui-def element.

3. For each report, specify values for the elements and attributes listed in Production Reports Definition Elements on page 9-24.

4. When you have finished configuring each report, save and close the file.

Configuring Activity Reports

A sub-screen on the Reports page is defined for all of the activity reports. Use this section to configure each activity report to display information as you want to view it. You can also specify which reports can be run from the MIDM. Following is an excerpt from the Activity report configuration stanza.

```
<subscreen>
   <enable>true</enable>
   <root-object>Person</root-object>
   <tab-name>Activity Report</tab-name>
   <report-name>Transaction Summary</report-name>
   <screen-id>5</screen-id>
   <display-order>4</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-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/>
      </search-result-list-page>
   </search-result-pages>
</subscreen>
```

To Configure Activity Reports

Perform the following steps for each production and activity report.

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **reports element**. This section is located near the end of the file.
3. In the subscreen element that defines the activity reports, modify any of the elements described in Activity Reports Page Definition Elements on page 9-33 to configure the Activity Reports page:
4. For each activity report, modify any of the elements described in Activity Reports Search Elements on page 9-33.
5. To specify the size and display of the results returned for the search, modify any of the elements described in Activity Reports Results Elements on page 9-33.
6. When you have finished configuring each report, save and close the file.

Production Reports Definition Elements

The following table lists and describes the elements you can configure in `midm.xml` to define the production reports.

Element/Attribute	Description
enable	An indicator of whether to display the report search page on the Reports page. Specify **true** to display the report search page and enable the report; specify **false** to disable the report.
tab-name	The name to display on the tabbed heading for the report.
report-name	The name of the report as it appears on the generated report.
display-order	An integer indicating the order in which each report tab appears on the Reports page. If you change the display order of one report, make sure to change the others to keep the numbers consecutive.
item-per-page	The number of items to display on one page of the report search results.
max-result-size	The number of records to display on the report. if no value is entered, or if the value is zero (0), the size defaults to 1000 records. To retrieve all records for a report, enter a very large value for this element.

Element/Attribute	Description
description	A description of the fields defined for the field-group element. This value appears as a 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 names of the fields 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.

Activity Reports Definition Elements

The following tables list and describe the elements you can configure for the activity reports in `midm.xml`.

Activity Reports Page Definition Elements

The following table lists and describes the elements you can configure in `midm.xml` to define the Activity Reports page.

Element/Attribute	Description
enable	An indicator of whether to display the Activity Reports search page on the Reports page. Specify **true** to display the report search page and enable the report; specify **false** to disable the report.
tab-name	The name to display on the tabbed heading for the report.
report-name	The name of the report as it appears on the generated report.
display-order	An integer indicating the order in which the Activity Reports tab appears on the Reports page. If you change the display order of one report, make sure to change the others to keep the numbers consecutive.

Activity Reports Search Elements

The following table lists and describes the elements you can configure in `midm.xml` to define each activity report search.

Element/Attribute	Description
screen-title	The name of the report search page as it appears on the Activity Reports page.
report-name	The name of the report as it appears on the generated report.
search-screen-order	An integer indicating the order in which each activity report appears on the Activity Reports page. If you change the display order of one report, make sure to change the others to keep the numbers consecutive.

Activity Reports Results Elements

The following table lists and describes the elements you can configure in `midm.xml` to define each activity report search result.

Element/Attribute	Description
item-per-page	The number of items to display on one page of the report search results.

Element/Attribute	Description
max-result-size	The number of records to display on the report. if no value is entered, or if the value is zero (0), the size defaults to 1000 records. To retrieve all records for a report, enter a very large value for this element.

Configuring the MIDM Source Record Page

The Source Record page is where an MIDM user adds new records to the master person index database and performs actions against a source record rather than an enterprise record. This page includes sub-screen configurations, which means there are a series of tabbed pages within the Source Record page. These pages each provide different functions, such as view and edit, compare and merge, and create a new record.

- Configuring the Source Record Page Definition on page 34
- Configuring the Tabbed Pages on the Source Record Page on page 34

Configuring the Source Record Page Definition

You can configure certain display options for the main Source Record page, including the order in which the tab appears on the MIDM and the name of the page.

To Configure the Source Record Page Definition

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **source-record element** (located in the gui-definition element).
3. To specify a different name for the Source Record tab, modify the value of the tab-name element.
4. To change the order in which the Source Record tab appears on the MIDM, change the value of the display-order element.

   Note: You might need to renumber the remaining tabs to keep the numbering sequential.

5. Save and close the file.

Configuring the Tabbed Pages on the Source Record Page

You can configure any of the pages on the Source Record page by changing the tab name and display order and by enabling or disabling each page. Do not modify the search page or search results list for the sub-screens. Below is a sample configuration for the View/Edit page on the Source Record page.

```
<subscreen>
   <enable>true</enable>
   <root-object>Person</root-object>
   <tab-name>View/Edit</tab-name>
   <screen-id>0</screen-id>
   <display-order>0</display-order>
   ...
```

You should never need to modify the root-object or screen-id elements.

To Configure the Source Record Tabbed Pages

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **source-records element**, and then to the subscreen element containing name of the tab you want to modify.
3. To specify a new name for the tab, modify the value of the tab-name element.
4. To disable the sub-screen, change the value of the enable element to **false**.
5. To enable the sub-screen, change the value of the enable element to **true**.
6. To change the order in which the tab appears on the Source Records page, change the value of the display-order element.

   Note: You might need to renumber the remaining tabs to keep the numbering sequential.

7. Save and close the file.

Configuring the MIDM Audit Log Page

When enabled, the audit log stores a history of each instance in which information from the object tables in the master person index database is accessed. The MIDM allows you to search for and view the audit log entries. You can enable or disable the audit log, change the display order, and customize the size of the results list. Below is the default Audit Log page definition.

```
<audit-log>
   <allow-insert>false</allow-insert>
   <root-object>Person</root-object>
   <tab-name>Audit Log</tab-name>
   <screen-id>7</screen-id>
   <display-order>7</display-order>
   <search-pages>
      <simple-search-page>
         <show-euid>true</show-euid>
         <show-lid>true</show-lid>
      </simple-search-page>
   </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>100</max-result-size>
         <field-group/>
      </search-result-list-page>
   </search-result-pages>
</audit-log>
```

To Configure the Audit Log Page

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **audit-log element** (located in the gui-definition element).
3. Make any of the following changes:
   - To specify that records be written to the audit log, change the value of the allow-insert element to **true**.
   - To specify that records not be written to the audit log, change the value of the allow-insert element to **false**.
   - To change the name of the Audit Log tab on the MIDM, change the value of the tab-name element.
   - To change the order in which the Audit Log appears on the MIDM, change the value of the display-order parameter to a **different integer**.

     Note: If you change the order of one tab, make sure to change the order of the remaining tabs so no tabs have an identical display order value.

   - To change the number of Audit Log search results that appear on one page, change the value of the item-per-page element.
   - To change the maximum number of audit log records that can be returned for a search, change the value of the max-result-size element.
4. Save and close the file.

Configuring Master Index Data Manager Implementation Information

Certain configuration information is defined automatically in the implementation details section of `midm.xml` based on information you specify in the wizard. Other information in this section must be customized by modifying the XML file directly.

You can customize the implementation details by performing any of the following tasks.

- Specifying the Master Controller JNDI Class on page 36
- Specifying the Master Person Index Report Generator JNDI Class on page 37
- Specifying Master Person Index Validation Services on page 37
- Setting Master Person Index Debug Options on page 38
- Specifying a Master Person Index Field Masking Class on page 38

Specifying the Master Controller JNDI Class

The MIDM must know the name of the Master Controller interface for the Oracle Healthcare Master Person Index implementation. Only change this value if you create a custom Master Controller.

To Specify the Master Controller JNDI Class

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.
2. Scroll to the **impl-details element**, and then to the **master-controller-jndi-name element**.

3. Modify the value of the master-controller-jndi-name element to the name of the Master Controller JNDI class for your server.

   For example:

   ```
   <master-controller-jndi-name>ejb/CustomerMasterController
   </master-controller-jndi-name>
   ```

4. Save and close the file.

Specifying the Master Person Index Report Generator JNDI Class

The MIDM must know the name of the class used to generate reports for the MIDM. Only change this value if you create a custom report generator.

To Specify the Report Generator JNDI Class

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the impl-details element, and then to the **reportgenerator-jndi-name element**.

3. Modify the value of the reportgenerator-jndi-name element to the name of the report generator JNDI class to use.

   For example:

   ```
   reportgenerator-jndi-name>ejb/CustomerReportGenerator
   </reportgenerator-jndi-name>
   ```

4. Save and close the file.

Specifying Master Person Index Validation Services

Validation services verify processing codes for data entered into the master person index database. Only change the names of the validation services if you created custom code list validators.

To Specify the Validation Service

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **impl-details element**, and then to the **validation-service-jndi-name element**.

3. To change the name of the validator for processing codes stored in sbyn_common_header, modify the value of the validation-service-jndi-name element to the name of the JNDI class used for validation.

   For example:

   ```
   <validation-service-jndi-name>ejb/CustomerCodeLookup
   </validation-service-jndi-name>
   ```

4. To change the name of the validator for processing codes stored in sbyn_user_code, modify the value of the usercode-jndi-name element to the name of the JNDI class used for validation.

For example:

```
<usercode-jndi-name>ejb/CustomerUserCodeLookup
</usercode-jndi-name>
```

5. Save and close the file.

Setting Master Person Index Debug Options

When you first implement Oracle Healthcare Master Person Index, you might want to view detailed debug information until you are certain the application is working as required. You can set debug options in the implementation details.

To Set Debug Options

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **impl-details element**, and then to the **debug-flag element**.

3. Modify the value of the debug-flag element to indicate whether you want debug information logged.

   For example:

   ```
   <debug-flag>true</debug-flag>
   ```

   Specify **true** to log debug information; specify **false** to turn logging off.

4. Modify the value of the debug-dest element to indicate where to print debug information.

   For example:

   ```
   <debug-dest>console</debug-dest>
   ```

   Specify **console** to log debug information to a monitor; specify **file** to print to a file.

5. Save and close the file.

Specifying a Master Person Index Field Masking Class

If you implement a custom class to define field masking logic, you must specify the new class in the implementation details. The custom class must be created as a custom plug-in in the Oracle Healthcare Master Person Index project. This class can be used to mask the value of any field for which the is-sensitive element is set to "true".

To Specify a Field Masking Class

1. In the Projects window, expand the Configuration node in the project you want to modify, and then double-click **midm.xml**.

   The file opens in the NetBeans XML editor.

2. Scroll to the **impl-details element**.

3. Change the value of the object-sensitive-plug-in-class element to the name of the custom class.

   For example:

```
<object-sensitive-plug-in-class>
 com.sun.mdm.index.user.customfieldmasker
</object-sensitive-plug-in-class>
```

4. Save and close the file.