Oracle® Healthcare Master Person Index

User's Guide

Release 4.0

E68430-01

February 2016

ORACLE®

Oracle Healthcare Master Person Index User's Guide Release 4.0

E68430-01

Contents

3   Oracle Healthcare Master Person Index Databases

5   Setting Up Application Servers to Use with Web Service

6   Web Layer Separation for OHMPI

7   Installing OHMPI on a WebLogic Cluster

8   Master Index Data Manager Login

9   Adding New Source Systems to an In-production MPI

Preface

This document introduces users to the Oracle Healthcare Master Person Index (OHMPI), providing conceptual information on the OHMPI development process, the runtime environment, and how to analyze business requirements prior to creating a master person index. It also provides instructions on how to create an OHMPI application and database.

Audience

This document is intended for anyone that is going to use the Oracle Healthcare Master Person Index.

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 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 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 Relationship Management Data Manager User's Guide*

- *Oracle Healthcare Master Person Index Relationship Management REST APIs Reference Guide*

- *Oracle Healthcare Master Person Index Relationship Management 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 Third Party Licenses and Notices*

- *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

Developing Oracle Healthcare Master Person Indexes

This chapter introduces you to the Oracle Healthcare Master Person Index (OHMPI), and provides an introduction to the development process. It also introduces you to the Master Index Data Manager (MIDM) object profiles, and provides information you should consider before developing a master person index, including a checklist.

This chapter includes the following sections:

- Understanding the Master Person Index Development Process on page 1-1
- Learning About the Master Person Index Framework and Runtime Environment on page 1-2
- Analyzing Your Business Requirements before Developing a Master Person Index on page 1-3

Understanding the Master Person Index Development Process

This document provides instructions for creating a master person index application using Oracle Healthcare Master Person Index. It includes creating the master person index application framework, creating custom Java code, generating the application files, and creating the database. It also includes performing a data analysis using the framework created for the master person index application.

The following steps outline and link to the procedures you need to follow to develop a master person index application. Not all tasks are covered in this document, and when they are covered elsewhere that guide is named.

1. Perform a preliminary analysis of the data you plan to store in the master person index application.

2. Create a new project and a new Oracle Healthcare Master Person Index application within that project (described in Creating a Master Person Index or IHE Profile Application on page 2-2).

3. Define the object structure, operating environment, and certain runtime characteristics using the wizard (described in Creating a Master Person Index or IHE Profile Application on page 2-2).

4. Define and build custom plug-ins, and specify the plug-ins in the appropriate configuration file (described in Implementing OHMPI Custom Plug-ins on page 2-20).

5. Customize the configuration files. For more information, see *Oracle Healthcare Master Person Index Configuration Guide*.

6. Generate the master person index application (described in Generating Master Person Index on page 2-21).

7. Create the database (described in Creating an Oracle Healthcare Master Person Index Database on page 3-9).

   ■ Customize the database scripts by defining system information, processing codes, and drop-down menu values.

   ■ Create the database structure and any necessary indexes.

   ■ Define database connectivity in the application server.

8. Analyze and cleanse existing data that will be pre-loaded into the master person index database (instructions for this step are provided in *Oracle Healthcare Master Person Index Analyzing and Cleansing Data User's Guide*).

9. Build and deploy the project.

10. Define security for the Master Index Data Manager (MIDM) described in *Oracle Healthcare Master Person Index Data Manager User's Guide*.

11. Load the initial data set into the master person index database (this is described in *Oracle Healthcare Master Person Index Loading the Initial Data Set*).

Learning About the Master Person Index Framework and Runtime Environment

The values you enter in the wizard, Configuration Editor, or directly in the XML files define how other components of the master person index application are generated, such as the database scripts, the Master Index Data Manager (MIDM), and the dynamic Java API. This section provides an overview of how the values you enter correspond to the runtime environment.

From XML to the Database

The master person index database is created using a standard Oracle database and a database script generated directly from `object.xml`. Additional scripts are created based on any user codes or menu lists you defined for the fields in the object structure. Running the database scripts against the database creates the tables necessary for your master person index application and also creates startup data, such as external system information or processing code.

From XML to the Master Index Data Manager

Based on information you specify in the wizard or Configuration Editor, `midm.xml` is generated to define the appearance of the Master Index Data Manager. This file defines the fields and appearance of the MIDM and also specifies the searches used by the MIDM. The available search types are defined in `query.xml`. You can customize many features of the MIDM, including the following:

■ The fields that appear on the MIDM pages

■ The field attributes, such as a display name, order of appearance, length, type, and data format

■ The types of searches that can be performed and the fields available for each type

■ The appearance of search results lists

■ The location of the fields on all windows

From XML to the EJB

When you generate the master person index application, two additional projects are created, an EJB project and a web application project. The EJB project includes several Java classes that are common to all master person index applications as well as a set of custom classes that are specific to the type of object you are indexing. These are based on `object.xml`. You can use these methods to specify how data is processed into the database. You can also call these methods from Business Processes.

From XML to the Runtime Environment

The information you specify in the master person index configuration files is read at runtime when the domain is started. The only exception is `object.xml`, which is stored only as a record of the object structure. You can modify the configuration files after moving to production; however, for the changes to take effect, you must regenerate the application and then rebuild and redeploy the project to apply the changes to the server.

> **Caution:** Use caution when modifying these files; changing these files after moving to production might result in loss of data integrity or unexpected weighting and matching results.

Analyzing Your Business Requirements before Developing a Master Person Index

Creating a master person index application requires in-depth analyses of your business requirements, legacy data, and data processing requirements. After the initial analysis, you can plan and design how you will create the master person index application framework and how you will customize that configuration after creating the framework.

The following sections provide information about what you need to know and do before you begin to create a master person index application.

- Preliminary Data Analysis for a Master Person Index on page 1-3
- Planning a Master Person Index Project on page 1-3
- Master Person Index Project Initiation Checklist on page 1-4

Preliminary Data Analysis for a Master Person Index

Before creating the master person index application, perform a preliminary analysis against the data that will be stored in the index database to determine the structure of the records. Analyzing your data requires reviewing the message structure of each legacy system that will share data with the master person index application. The master person index application does not need to store every field from every system, so you can narrow the master person index application object structure to just the pertinent fields from each system. However, the master person index application stores the same fields for each system. A more in-depth analysis of the field values in the legacy data occurs after the initial master person index application framework is created.

Planning a Master Person Index Project

Before you create the Oracle Healthcare Master Person Index project, analyze the business requirements and determine which project components will help you meet those requirements. Planning the project includes defining how each external system will share information with the master person index application and how the master

person index application will share information with those external systems. In addition, you can incorporate master person index Java methods that define how the master person index application processes incoming data. Master person index methods can also be used to transform the data sent from external systems into a format that can be read by the master person index application.

An additional consideration is whether to integrate the master person index methods into a Business Process.

Master Person Index Project Initiation Checklist

Before you begin developing your master person index application, make sure you have obtained the following information:

- The primary object to be indexed, such as a person, customer, or business

- Any secondary objects, such as telephone numbers and addresses

- All fields to be stored in the index for both the primary and secondary objects

- The name of each field as it appears on the MIDM and whether the field will be a standard text field or will be populated from a menu list (if a field will be populated from a menu list, you should also define an eight-character name for the list)

- The fields that are required in order to add a record or that are required for queries

- The fields that will appear on reports

- The fields that must be unique to an enterprise record (in other words, they uniquely identify a child object within an enterprise record)

- The fields that will be used for matching

- The fields that will need to be parsed or normalized prior to matching

- Any special formatting requirements, such as character types, the data type, minimum and maximum values, and field size

- The fields that will appear on MIDM search and search results windows

- The processing codes for the source systems being integrated into the index

2

Master Person Index and IHE Profile Project Creation

This chapter provides information and a procedure for creating an Oracle Healthcare Master Person Index application. It also provides information about master person index wizard field properties and name restrictions, and custom plug-ins, including how to implement custom plug-ins, and generate a master person index.

This chapter includes the following sections:

- Setting Your System for Languages Other Than English to Display Properly on page 2-1

- Creating a Master Person Index or IHE Profile Application on page 2-2

- Understanding OHMPI Wizard Name Restrictions and Field Properties on page 2-10

- Using Custom Plug-ins for Custom Transaction Processing on page 2-14

- Learning About Custom Plug-Ins for OHMPI Custom Components on page 2-17

- Implementing OHMPI Custom Plug-ins on page 2-20

- Generating Master Person Index on page 2-21

- Upgrading 3.x Master Index Application to 4.0 Release on page 2-23

Setting Your System for Languages Other Than English to Display Properly

This section provides information for setting your system to display languages other than English properly.

- Setting the Language and Region if Different Than English and United States on page 2-1

- Setting the UTF-8 Encoding in NetBeans IDE for Simplified Chinese or Languages with Accents on page 2-2

Setting the Language and Region if Different Than English and United States

If you plan to use a language other than United States English, set your computer to the desired language and country. For example, if you are using:

- Spanish in Mexico, select **Mexico**

- Simplified Chinese in the People's Republic of China, select **China**

- Japanese in Japan, select **Japan**

To Change Regional and Language Options Using Microsoft Windows

> **Note:** If using an operating system other than Windows, see your computer system's manual.

1. Click **Start**, select **Control Panel**, and then double-click **Regional and Language Options**.

   The Regional and Language Options window appears. In our example, we presume you want to use Simplified Chinese.

2. Click the **Regional Options** tab.

3. In the Location section, select **China** from the drop-down list.

4. Click **OK**.

When your computer system is set to Chinese language and you create an OHMPI project, a copy of `midm_person_chinese.xml` will be created in the `project_dir/src/Configuration` folder.

> **Note:** This file (`midm_person_chinese.xml`) is based upon the Person template and is translated into Chinese. For projects created with the Person template, you can replace the `midm.xml` with this file.

Setting the UTF-8 Encoding in NetBeans IDE for Simplified Chinese or Languages with Accents

For languages with accents (Simplified Chinese, Spanish, or Japanese) to display properly in NetBeans, you must add "-J-Dfile.encoding=UTF-8" to the end of "netbeans_default_options" in *OracleHealthcareMPIv4_0*\netbeans\etc\netbeans.conf. For example:

```
netbeans_default_options="-J-Dnb.registration.enabled=\"false\" -J-Xmx512m
-J-client -J-Xss2m -J-Xms512m -J-XX:PermSize=32m
-J-Dapple.laf.useScreenMenuBar=true -J-Dapple.awt.graphics.UseQuartz=true
-J-Dsun.java2d.noddraw=true -J-Dsun.java2d.dpiaware=true
-J-Dsun.zip.disableMemoryMapping=true -J-Dfile.encoding=UTF-8"
```

You must set this option prior to using NetBeans.

Creating a Master Person Index or IHE Profile Application

The wizard provides a simple method for you to create the object definition and the runtime configuration files for your master person index application. This section provides instructions for creating a new project and using the wizard to create the configuration files for the master person index application. To create the initial master person index framework, follow these steps:

Step 1: Choose Project

When you create a new Oracle Healthcare Master Person Index project in NetBeans, the wizard for creating a master person index application automatically appears.

To Choose a Project

1. On the NetBeans toolbar, click **New Project**.

   The New Project wizard appears, and opens on the Choose Project window.

2. Under Categories, select **OHMPI**.

3. Under Projects, select **Master Person Index Application** or **IHE Profile Application** and then click **Next**.

   > **Note:** In our example, we selected Master Person Index Application.

   The Name and Location window appears with default project name and directory paths.

4. Continue to Step 2: Name and Location.

Step 2: Name and Location

When you create a new Oracle Healthcare Master Person Index project in NetBeans, the wizard for creating a master person index application automatically appears.

To Set the Name and Location

1. Enter the NetBeans project name and the path where you want to store the project files in the upper portion of the window or accept the defaults.

2. Select the application server you are using from the drop down list.

   a. If no application servers appear in the list, click **Manage**.

      The Servers window appears.

   b. Click **Add Server**.

      The Add Servers Instance window appears.

   c. Select the Oracle WebLogic Server, and click **Next**.

   d. Under Server Location, accept the defaults and click **Finish**.

      You are returned to the Servers window with the application server you selected listed under Servers in the left pane.

   e. Click **Close**.

      The Name and Location window reappears.

3. *For IHE Profile only*: Select **Oracle** from the drop-down list.

4. For IHE Profile application, the HL7 V2 Service Port is populated automatically (the default is **4447**. If desired, change the port number.

5. The **Set as Main Project** is selected for the Master Person Index application but not for the IHE Profile Application. If you do not want this Master Person Index application to be the main project, deselect the option, and if you want the IHE Profile application to be the main project, select this option.

6. Click **Next**.

   ■ For an IHE Profile application, this is the final step that you need to perform.

   ■ For a Master Person Index application, continue to Step 3: Name Application.

Step 3: Name Application

Before you can configure the new master person index application, it is a good practice to give each master person index application a unique name because certain components within the application server are based on this application name. Dependent components include the data source for the JDBC connection and the outbound Topic (if one is used).

To Name the Master Person Index Application

1. Complete Step 2: Name and Location.

2. In the **Application** field of the Name Application window, enter a name for the new master person index application, and then click **Next**.

   > **Note:** Give the application the same name you will give to the parent object when you define the object structure.

   The Define Source Systems window appears.

3. Continue to Step 4: Define Source Systems.

Step 4: Define Source Systems

After you specify a name for the new master person index application, you need to specify the processing codes for the source systems that will be integrated into the master person index system.

The source system is the key to your network. If it a hospital (for example, Hospital or Medical Center, the system might be HOS or MC), all of its patient records would be sent into the master person index with a system code/LID of HOS/120210210 or MC/48481234.

The LIDs are specific to the patients. When you have to create data from the Master Index Data Manager (MIDM), every record you enter will need a System/LID. When you click the **System Record** tab of MIDM (see *Oracle Healthcare Master Person Index Data Manager User's Guide* for details) and click the **Add** tab, a drop-down list will have all of the hospital codes you entered. Now, supply the LID for the patient you are entering. If the data comes from the back end, it will have both System and LID as part of the data.

To Define Source Systems

1. Complete Step 3: Name Application.

2. In the **Name** field of the Define Source Systems window, enter the processing code of one of the source systems that will share data in the master person index system (for example, "GNH" or "MCT"), and then click **Add**.

   You can enter any of the following characters:

   - ! _ ~ ( ) { } + \Q # $ % & : ; - /
   - a-z
   - A-Z
   - 0-9
   - þ ÿ Þ ß 'à-ö ø-ý À-Ö Ø-Ý

   The value you entered appears in the Systems box.

   > **Note:** Be sure to enter the processing code of the system and not the name. This value is entered in update.xml for defining survivor strategies for the single best record (SBR).

3. Do any of the following:

   - To define additional systems, repeat the above step for each source system that will share information with the master person index application.
   - To remove a system from the list, highlight the name of that system in the Systems box, and then click **Remove**.
   - To remove all systems from the list, click **Remove All**.

4. When you have defined all required source systems, click **Next**.

   The Define Deployment Environment window appears.

5. Continue to Step 5: Define Deployment Environment.

Step 5: Define Deployment Environment

Once you define the source systems, you must specify information about the deployment environment, including the database and match engine vendors.

To Define the Deployment Environment

1. Complete Step 4: Define Source Systems.

2. On the Define Deployment Environment window, enter the following information:

   - **Database** - The type of database being used for the master person index application. Select **Oracle**.
   - **Date and Time Formats** - The date and time formats for the master person index system. This defines how date and time must be entered and how they appear on the MIDM. These formats acts as a default format if you do not set the field level format.
     - **Object Date Format**: This is the default or application level format used by OHMPI backend for the *date* data type.
     - **Object Time Format**: This is the default or application level format used by OHMPI backend for the *timestamp* data type.

- **MIDM Date Format**: This is the default or application level format used by MIDM for the *date* data type.

- **MIDM Time Format**: This is the default or application level format used by MIDM for the *timestamp* data type.

■ **Transaction Mode** - An indicator of whether transactions are distributed. Select one of the following options:

- **XA (across applications)**, if the transactions will be distributed across multiple applications.

- **XA (within application only)**, if the transactions will be distributed just across the master person index application.

- **Local**, if the transactions will not be distributed.

3. Click **Next**.

4. Continue to Step 6: Define Enterprise Object.

Step 6: Define Enterprise Object

After you define the deployment environment for the master person index application, you can begin to define the structure of the object you want to index. The primary object will be the parent object for any other objects defined. Child objects are not required if all information is stored under the parent object.

You can create new user-defined objects, create objects using predefined templates, or use a combination of both methods to create the objects in your enterprise object. Perform any of the following actions to define the objects in the enterprise object.

■ Creating User-defined Objects on page 2-6

■ Creating Objects from a Template on page 2-9

■ Deleting an Object from the Structure on page 2-9

Complete Step 5: Define Deployment Environment before performing these procedures.

> **Note:** The names of database constraints are created based on the parent and child object names. Due to database naming restrictions, the length of the parent object name plus the length of any of the child object names *must be 21 characters or less*. Give the parent object the same name you gave the application earlier in the wizard.

Creating User-defined Objects

When you create user-defined objects, you create an empty object with no predefined fields or child objects.

1. On the Define Enterprise Object window, click the **Add Primary Object** icon.

The initial node appears on the tree. By default, the name of the parent object is the same as the name of the application you defined in Step 3: Name Application.

2. Accept the default name by pressing **ENTER**, or type a new name and press **ENTER**.

> **Note:** Oracle recommends that when you define the object structure that the name of the parent object be the same as the application name you created in step 3.

3. To create a new child object, select the primary object created above and then click the **Add Sub Object** icon.

   The new child node appears on the tree.

4. Highlight the new sub-object and type a user-defined name for it and press **ENTER**.

5. Repeat the previous two steps for each child object.

6. Continue to Defining the Fields for Each Object.

> **WARNING:** 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 the Fields for Each Object

After you define all the parent and child objects for your enterprise object, you need to define the fields in each object. Every field has a set of properties that must be configured before creating the master person index configuration files. If you chose a predefined template to create your objects, be sure to check the properties for all predefined fields to be sure they are configured correctly for your implementation.

> **Note:** If you decide that you would like to use a template as opposed to creating everything from scratch, highlight your primary object and click the **X** in the tool bar. This deletes the primary object you are working on. You can select the **Add Solution** icon, and perform one of the following:
>
> - Highlight **Generic** and then select **Person** or **Company**
>
> - Highlight **Provider** and then select **Organization** or **Individual**
>
> - Highlight **Patient** and then select **USPatient**, **UKPatient**, or **AUPatient**
>
> See Creating Objects from a Template on page 2-9 for additional information.

> **Caution:** Delete primary objects, sub-objects, and fields carefully, for once deleted you cannot retrieve them.

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

Adding a Field

After you define all the parent and child objects for your enterprise object, you need to define the fields in each object. Every field has a set of properties that must be configured before creating the master person index configuration files. If you chose a predefined template to create your objects, be sure to check the properties for all predefined fields to be sure they are configured correctly for your implementation.

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

1. Complete Creating User-defined Objects on page 2-6.

2. In the object tree panel of the Define Enterprise Object window, perform 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**.

   - 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**.

     The tree expands and a new field is inserted.

3. Perform one of the following:

   - To accept the default name, press **ENTER**.

   - To change the name, type the new name and then press **ENTER**.

     For information about field name restrictions, see OHMPI Wizard Field Name Restrictions on page 2-11.

4. Continue to Configuring Field Properties.

Configuring Field Properties

When you create a field, a set of default properties is defined for that field. You can modify the property configuration for each field to suit your data processing, storage,

and display requirements. After you modify a property value, press **ENTER** to apply the change.

> **Note:** If you do not press **ENTER**, you will lose the name you typed.

1. Complete the steps under Adding a Field on page 2-8.

2. In the object tree panel of the Define Enterprise Object window, select the field you want to configure.

3. On the Properties panel in the right side of the window, modify the value of any of the properties listed in Master Person Index Wizard General Field Properties.

4. On the right side of the window, click the **MIDM** tab, and then modify the value of any of the properties listed in Master Person Index Wizard MIDM Field Properties.

5. When you have created and configured all of the necessary fields for each object, click **Next**.

6. Continue to Step 7: Generate the Project Files.

Creating Objects from a Template

When you create objects from a template, secondary objects and fields are predefined.

1. To create a complete object structure from a template, click the **Add Solution** icon on the Define Enterprise Object toolbar, and perform one of the following:

   - Highlight **Generic** and then select **Person** or **Company**

   - Highlight **Provider** and then select **Organization** or **Individual**

   - Highlight **Patient** and then select **USPatient**, **UKPatient**, or **AUPatient**

   For more details about the Provider and the Patient template solutions, see the following guides: *Oracle Healthcare Master Person Index Provider Index User's Guide*, *Oracle Healthcare Master Person Index United States Patient Solution User's Guide*, *Oracle Healthcare Master Person Index United Kingdom Patient Solution User's Guide*, or the *Oracle Healthcare Master Person Index Australia Patient Solution User's Guide*.

   The Provider and the Patient template solutions are pre-set and you cannot edit or configure them until after you have generated the application file.

Deleting an Object from the Structure

If you add an object in error, or do not want to use one of the objects in a predefined template, you can delete the object from the structure.

1. On the Define Enterprise Object window, select the object you want to remove.

2. Do any of the following:

   - Right-click in the object tree panel, and then select **Delete**.

   - Press the **Delete** key.

   - In the wizard toolbar, click **Delete**.

     The object and any fields associated with that object are deleted. If you remove the parent object, all child objects are deleted.

> **Note:** Once you delete an object, it is gone and you cannot retrieve
> them.

Step 7: Generate the Project Files

Once you have named the application and configured the source systems, deployment environment, objects, and fields for the master person index application, you need to generate the configuration files and database scripts. Review the configuration files to be sure the application is set up correctly for your data processing environment.

> **Note:** Modifying the configuration files is not covered in this
> document. For instructions on configuring a master person index
> application, see *Oracle Healthcare Master Person Index Configuration
> Guide*.

To Generate the Configuration Files

1. Complete Step 6: Define Enterprise Object.

2. On the Generate Project Files window, verify that all of the information you have entered is complete and correct.

3. If desired, generate the remaining Master Person Index application files by selecting the appropriate box, and then click **Finish**.

> **Note:** Although you can generate the remaining Master Person
> Index application files at this time, it is not necessary to do this now as
> you can also generate them when your project appears in NetBeans.
> Waiting may be the better option, for then you can configure your files
> in NetBeans before generating them. If you generate them at this time,
> you will have to generate them a second time after you complete the
> configuration process.

The configuration files are generated, and are stored in the project configuration folder.

4. Continue to Step 8: Review the Configuration Files.

Step 8: Review the Configuration Files

After the wizard is complete, several nodes representing the master person index configuration files are placed in the project for the master person index application. Verify that the configuration files are customized correctly for your implementation. If you need to modify the configuration files, see *Oracle Healthcare Master Person Index Configuration Guide* for information about configuration tasks. For information about the configuration files and configurable options, see *Oracle Healthcare Master Person Index Configuration Reference*.

Understanding OHMPI Wizard Name Restrictions and Field Properties

When you create fields in the object structure of the master person index application, you can specify several properties for each field, such as whether the field is required, whether the field will appear as a drop-down menu on the MIDM, or whether the field

will be used in a blocking query. There are also some restrictions for how fields can be named and how the properties are defined.

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

OHMPI Wizard 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.

■ In field names, do not use characters or names restricted by Java, XML, or the database platform being used.

OHMPI Wizard General Field Properties

The following table lists and describes the properties you can define on the General Properties page of the wizard.

Property	Description
Data Type	The master person index data type of the field. The following data types are supported: ■ **string** - Contains a string of characters. ■ **date** - Contains a date value. ■ **float** - Contains a floating point integer. ■ **int** - Contains an integer. ■ **char**[1] - Contains a single character. ■ **boolean**[2] - Contains either **true** or **false**. ■ **blob**[3] - Contains binary data.
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. 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 wizard match types. See *Oracle Healthcare Master Person Index Configuration Reference* for information about the available options for this field and how the wizard match types correlate to mefa.xml types.
Blocking	An indicator of whether the field will be used in the blocking query. Specify **true** to add the field to the blocking query; specify **false** to omit it from the blocking query.

Property	Description
Key Type	An indicator of whether the field is used to identify unique objects. For example, a business index might store several addresses for each business. Each address is assigned an address type and each business can only have one address of each type. Specify **true** if the field is a unique record identifier, or **false** if it is not.
	Key type fields should also be required fields (see below), unless a combination of fields are specified as key types for an object.
	Note: Oracle recommends that each child object contain a key type field, but this is not required. If child objects do not contain one or more key type fields, each enterprise object might accumulate a very large number of child objects depending on the survivor strategy used.
Updateable	An indicator of whether the field can be updated from the MIDM and external system messages. Specify **true** if the field can be updated or **false** if it cannot.
Required	An indicator of whether the field is required in order to save an enterprise object to the database. Specify **true** if the field is required or **false** if it is not. If only one key type field is defined for an object, that field should be required.
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.
	Labeled "Field Size" in the MIDM, it is "<size>" in the XML files.
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 Java. 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 in the following table. If no input mask is specified, all regular expression patterns are supported.
Code Module	The identification code for the drop-down list that appears for this field in the MIDM.
	Note: This value must match an entry in the code column of the `sbyn_common_header` database table and, by default, an entry for the code you enter is created in the Code List database script. You can further customize code lists in the script after completing the wizard.
User Code	The processing code for the drop-down list that appears for the fields defined by the Constrained By property. For more information, see the description of the Constrained By property below.
	Note: This must match an entry in the `code_list` column of the `sbyn_user_code` database table.
Constrained By	The name of the field that contains the corresponding User Code value (described above). The User Code and Constrained By properties are used in conjunction to fulfill two purposes. 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 that has a User Code value of CREDCARD, which is also 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. You could then create a field named ID that would be constrained by the formats defined for the ID Type field. Any IDs you enter would be validated against the value of the ID Type field.

¹ If you select char when working in the OHMPI wizard, the NetBeans Editor, or the Configuration Editor, this selection will limit the number of characters a field can contain to 1 character (for example, the Field Size is now limited to a 1 character entry).

² If you select boolean when working in the OHMPI wizard, the NetBeans Editor, or the Configuration Editor, this selection will limit you to two one-byte values in certain fields. For example, fields that require a "True" or "False" response will now be restricted to a "T" or "F" response.

³ If you select blob when working in the OHMPI wizard, the NetBeans Editor, or the Configuration Editor, this selection will gray out all the other fields, preventing you from setting any values in them. However, in the database, all these fields will now have unlimited space for any type of free-form entry. Be aware that there are two limiting factors when you choose blob: 1) the MIDM will not be available to view your entries in the fields, which will never appear in the MIDM, and 2) for you to view or change the values of the affected fields you must do so by accessing your database from the back end.

Master Person Index Wizard MIDM Field Properties

The following table lists and describes the field properties you can define on the MIDM Field Properties page of the wizard.

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, enter 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**. ■ **D** - indicates a numeric character. ■ **L** - indicates an alphabetic character. ■ **A** - indicates an alphanumeric character. Note that the value you enter can further restrict the data pattern for the field (this is the Pattern field property described in OHMPI Wizard General Field Properties on page 2-11).
Value Mask	A mask used by the index to strip any extra characters that were added by the input mask (see above). This mask ensures that data is stored in the database in the correct format. To specify a value mask, type the same value as is entered for the input mask, but type an "x" in place of each punctuation mark. For example, if an SSN field has an input mask of *DDD-DD-DDDD*, specify a value mask of **DDDxDDxDDDD** to strip the dashes before storing the SSN. A value mask is not required for date fields.
Search Screen	An indicator of whether the field appears on the search windows of the MIDM. Specify **true** to display the field or **false** to hide it.
Required	An indicator of whether the field must be populated on the search windows of the MIDM when performing a search. This property can only be modified if the Search Screen property is set to **true** (see above). Specify **true** to make the field required or specify **false** to make it optional. You can also specify **oneof** to create a group of fields of which at least one must be populated to perform a search. (Be sure to specify **oneof** for each field in the group.) Tip: If a field is required for a search, the field should also be required for creating a record (by specifying **true** for the Required property on the Properties page). Otherwise, searches performed from the MIDM could result in no matches even though possible matches exist.
Search Result	An indicator of whether the field appears on the search results windows of the MIDM. Specify **true** to display the field, or **false** to hide it.

Property	Description
Report	An indicator of whether the field appears on the reports generated from the MIDM. Specify **true** to display the field on the reports; otherwise specify **false**.

Using Custom Plug-ins for Custom Transaction Processing

You can add custom processing to the master person index application by adding your own code to new Java classes in the EJB project for the master person index application. These classes are called custom plug-ins. This ability allows you to tailor how messages are processed by the master person index application. Plug-ins can be used to customize field validations, update policies, match processing logic, and record retrieval, and to create custom components for the master person index application, such as custom phonetic encoders, block pickers, or query builders. You can create as many classes as you need to carry out the custom processes.

The following sections describe custom plug-ins that define custom processing. These are explained more fully in *Oracle Healthcare Master Person Index Configuration Reference*.

- OHMPI Update Policy Plug-ins define custom processing logic to perform against the resulting record of a transaction before it is stored in the database.

- OHMPI Field Validation Plug-ins define validations to perform against specific fields, such as checking the local ID length and format.

- OHMPI Field Masking Plug-ins define how the values for sensitive fields are hidden on the MIDM from users who do not have permission to view them.

- OHMPI Processing Logic Plug-ins define custom logic based on predefined decision points for how records are matched during a transaction.

- OHMPI Custom Plug-in Exception Processing defines how exceptions are handled by the custom plug-ins you create.

OHMPI Update Policy Plug-ins

For the primary transactions performed by the master person index application, you can define additional custom processing to perform against the record that results from a transaction. The policies you define are invoked by the Update Manager and are applied to the resulting records after they are processed by the survivor calculator. The modifications made to a record by an update policy determine how the record is stored in the database. By creating custom plug-ins, you can create additional Java classes to support the update policies you define.

Update policies are specified in the `UpdatePolicy` section of `update.xml`, and there are several different types. Each policy modifies an enterprise object (`class com.sun.mdm.index.objects.EnterpriseObject`) and must implement `com.sun.mdm.index.update.UpdatePolicy`, which contains one method, `applyUpdatePolicy`. The syntax is as follows:

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

This method throws two exceptions:

- `com.sun.mdm.index.master.UserException`

- `com.sun.mdm.index.objects.exception.ObjectException`

Enterprise Merge Policy

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

Enterprise Unmerge Policy

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

Enterprise Update Policy

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

Enterprise Create Policy

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

System Merge Policy

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

System Unmerge Policy

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

Undo Assumed Match Policy

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

OHMPI Field Validation Plug-ins

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

■   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`.

One default field validator, *validate-local-id*, is provided to validate system and local ID fields before processing data into the database. This is described in *Oracle Healthcare Master Person Index Configuration Reference* (E18592-01).

OHMPI Field Masking Plug-ins

There might be instances where you want to mask certain data in records from general users of the Master Index Data Manager and only allow access by administrators. To do this, you can create a custom plug-in that displays asterisks (or other symbols) in place of the field values on the MIDM. Once you define the custom plug-in, specify the name of the custom plug-in Java class in the `object-sensitive-plug-in-class` element of `midm.xml`).

OHMPI Processing Logic Plug-ins

You can implement custom plug-ins that customize the way the execute match methods process data into the master person index application. When a record is entered into the master index system, match processing is performed by calling one of the following "execute match" functions from the `MasterController` class.

- `executeMatch`

- `executeMatchUpdate`

- `executeMatchDupRecalc`

- `executeMatchUpdateDupRecalc`

- `executeMatchGui` (this method is only called by the MIDM)

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

Custom Match Processing Logic Methods

There are five decision branches where custom logic can be inserted. At specific points in match processing, the execute match methods look for the value of the methods listed below. For more information about the methods, see the Javadocs for the master person index. These methods are contained in the `ExecuteMatchLogics` class in the package `com.sun.mdm.index.master`. For information about where the decision points are reached in the processing logic and how to change the logic, see *Oracle Healthcare Master Person Index Message Processing Reference*. The following methods specify the custom logic.

- `bypassMatching` - Indicates whether to perform the match process on incoming records or to bypass the match process.

- `disallowAdd` - Indicates whether an incoming message can be inserted as a new record.

- `disallowUpdate` - Indicates whether an incoming record can update an existing record.

- `rejectAssumedMatch` - Indicates whether to accept or reject an assumed match of two records.

- `rejectUpdate` - Indicates whether to accept or reject an update to an existing record.

Custom Match Processing Logic Plug-in Requirements

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

- `com.sun.mdm.index.objects.SystemObject`

- `com.sun.mdm.index.objects.EnterpriseObject`

- `com.sun.mdm.index.objects.exception.ObjectException`

- `com.sun.mdm.index.master.ExecuteMatchLogics`

- `com.sun.mdm.index.master.CustomizationException`

Custom Match Processing Configuration

If you create a custom plug-in that defines custom processing logic for the execute match methods, you must specify those custom plug-ins in `master.xml` in the master index project. If you create a plug-in for customizing logic in the execute match methods used by the back-end, specify the name of that class in the `logic-class` element. If you create a plug-in for the MIDM, specify the name of that class in the `logic-class-gui` element. For example:

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

For more information about `master.xml`, see the *Oracle Healthcare Master Person Index Configuration Reference*.

OHMPI Custom Plug-in Exception Processing

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

For more information about these exception classes, see the Javadocs for Oracle Healthcare Master Person Index.

Learning About Custom Plug-Ins for OHMPI Custom Components

Oracle Healthcare Master Person Index provides a flexible framework, allowing you to create custom Java classes to plug in to most master person index application components. The following sections provide descriptions of some components for which you can create custom classes to use in your master person index application.

- OHMPI Survivor Calculator Plug-ins define how the single best record (SBR) is generated.

- OHMPI Query Builder Plug-ins define how queries are performed against the master person index database.

- OHMPI Block Picker Plug-ins define how the blocks of fields in a blocking query are selected during a query.

- OHMPI Pass Controller Plug-ins define how the master person index application determines whether to perform additional match passes against a record.

- Match Engine Plug-ins define logic that connects to a match engine other than the OHMPI Match Engine.

- Standardization Engine Plug-ins define logic that connects to a standardization engine other than that provided by the Master Person Index Standardization Engine.

- Phonetic Encoders Plug-ins for an OHMPI define logic that connects to phonetic encoders other than those provided.

OHMPI Survivor Calculator Plug-ins

The survivor calculator determines which field values from the various system records will populate the SBR for the enterprise record. You can create a custom survivor calculator class that selects the surviving field values. Your custom class must implement the survivor calculator interface. Call `selectField` in `com.sun.mdm.index.survivor.SurvivorStrategyInterface` to return the SBR value for each field. For more information about the classes and methods to use, see the Javadocs for Oracle Healthcare Master Person Index. The primary classes are contained in the `com.sun.mdm.index.survivor` package. Enter the fully qualified class path for the custom survivor calculator in `update.xml`.

OHMPI Query Builder Plug-ins

The query builder defines the different types of queries that can be used in the master person index application. You can implement custom queries using custom plug-ins. To create a new query builder, you must define a class that extends the base abstract `com.sun.mdm.index.querybuilder.QueryBuilder` and then specify that class in a query-builder element in `query.xml`. The exception thrown is `com.sun.mdm.index.querybuilder.QueryBuilderException`. The following methods must be implemented.

- `init` - This method receives the XML elements after the config element of `query.xml` so the query builder can read its custom configuration.

- `getApplicableQueryIds` - This method returns an array of string IDs indicating the query objects that can be generated given the available criteria. For example, in the blocking configuration, the unique ID of each block definition is the string that is returned by `getApplicableQueryIds`.

- `buildQueryObject` - This method constructs the query object based on one of the applicable query IDs provided as an input argument.

For more information about query-related Java classes, see the OHMPI Javadocs.

OHMPI Block Picker Plug-ins

The block picker chooses which block definition in the blocking query to use for the next matching pass. You can create a custom block picker class to select query blocks in a customized manner. If you create a custom block picker, specify the fully qualified

name of this custom plug-in for the block-picker element of `mefa.xml`. Follow these guidelines when implementing a custom block picker.

- Implement the `com.sun.mdm.index.matching.BlockPicker` interface to select the blocks in the desired order.

- If none of the remaining blocks should be executed, throw a `NoBlockApplicableException` from the `pickBlock` method.

OHMPI Pass Controller Plug-ins

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

- Implement the `com.sun.mdm.index.matching.PassController` interface to evaluate whether to do another pass or not.

- Return **true** from `evalAnotherPass` to specify that an additional pass be performed; return **false** to specify that no additional passes should be performed.

Match Engine Plug-ins

You can define classes to connect to a custom match engine instead of the OHMPI Match Engine. Specify the names of the custom classes you create in the `matcher-api` and `matcher-config` elements of `mefa.xml`. Follow these guidelines when implementing custom match engine classes.

- Implement the `com.sun.mdm.index.matching.MatcherAPI` interface to communicate with the match engine.

- Implement the `com.sun.mdm.index.matching.MatchEngineConfiguration` interface to retrieve any configuration values the match engine requires for initialization.

You can also define custom comparison functions to plug in to the OHMPI Match Engine. For more information, see *Oracle Healthcare Master Person Index Match Engine Reference* (E).

Standardization Engine Plug-ins

You can define classes to connect to a custom standardization engine instead of the OHMPI Standardization Engine. Specify the names of the custom classes you create in the `standardizer-api` and `standardizer-config` elements of `mefa.xml`. Follow these guidelines when implementing custom standardization engine classes.

- Implement the `com.sun.mdm.index.matching.StandardizerAPI` interface to communicate with the standardization engine.

- Implement the `com.sun.mdm.index.matching.StandardizerEngineConfiguration` interface to retrieve any configuration values the standardization engine requires for initialization.

You can also define custom standardization rules to plug in to the OHMPI Standardization Engine. For more information, see *Oracle Healthcare Master Person Index Standardization Engine Reference*.

Phonetic Encoders Plug-ins for an OHMPI

The master person index application supports several phonetic encoders, and you can define custom classes to implement additional phonetic encoders if needed. Specify the names of the custom classes you create in the *encoder-implementation-class* element of `mefa.xml`. When creating a custom phonetic encoder class, implement the `com.sun.mdm.index.phonetic.PhoneticEncoder` interface.

Implementing OHMPI Custom Plug-ins

Custom plug-ins allow you to incorporate custom code into your master person index application. When you create a custom plug-in for a master person index application, you define the package and class names. To incorporate the plug-ins into the master index application, you then specify the fully qualified class name of the class in the appropriate configuration file. For example, if you create a custom plug-in named `MergePolicy` in the package `com.sun.mdm.index.update`, the value to enter for the class in `update.xml` is `com.sun.mdm.index.update.MergePolicy`.

> **Note:** You can create custom plug-ins that define custom processing or that define custom components. For additional information about how to implement specific custom plug-ins, see Using Custom Plug-ins for Custom Transaction Processing and Learning About Custom Plug-Ins for OHMPI Custom Components.

You create a custom plug-in by composing Java code in a source file using the NetBeans editor. Create the file in the Source Package folder of the EJB project associated with the master person index application. Before you begin, determine the name of the package and class for the plug-in.

To Create Custom Plug-ins

1. In the NetBeans Projects window, expand *project_name*-ejb (where *project_name* is the name of the main Master Person Index Application or the IHE Profile Application project.

2. To create a new package for the custom plug-in files, do the following:

    a. Right-click **Source Packages**, point to **New**, and then click **Java Package**.

    b. Enter the package name and location, and then click **Finish**.

3. To create the plug-in, do the following for each source file:

    a. Under Source Packages, right-click the name of the package to which the new class will belong.

    b. Point to **New**, and then click **Java Class**.

    c. Enter the class name, verify the remaining fields, and then click **Finish**.

    d. Create the custom processing rules using Java code.

    e. When you are finished coding, save and close the file.

    f. In the Projects window, right-click the file and then click **Compile Package**.

        The output and status of the compiling process appears in the bottom panel.

> **Note:** If you do not compile a custom plug-in file, building the master person index project will compile the files.

4. Specify the name of the custom plug-in in the appropriate master person index configuration file.

> **Note:** If you modify a custom plug-in, be sure to recompile the file and regenerate the application to incorporate the changes. If the project has already been built, rebuild the project to add the custom plug-in to the EJB JAR file.

Generating Master Person Index

Before you generate your Master Person Index project, review the configuration files and make any necessary modifications. See *Oracle Healthcare Master Person Index Configuration Guide* (E25247-01) for more information. Once all modifications to the configuration files are complete and any custom plug-ins are built, generate your Master Person Index or IHE Profile project to create or update the project components. If you modify any of the configuration files, match and standardization engine files, or custom plug-ins after you generate the project, you need to regenerate the project to update the custom components.

> **Note:** If any errors occur while compiling the project, they appear in the output panel in the bottom of the window. This window also displays the status of the generate process.

To Generate the Project for the First Time

1. Save any configuration changes to the Master Person Index or the IHE Profile project.

> **Note:** While you can perform many configuration changes on a Master Person Index project, you are limited with the number of changes you can perform on an IHE Profile project.

2. Right-click the Master Person Index or the IHE Profile project in the Projects pane and then select **Clean and Build**.

   The project files are cleaned and built. This might take a few minutes.

3. Right-click the Master Person Index or the IHE Profile project in the Projects pane and then select **Generate Master Index Files** or **Generate Table Compression Scripts**. For more information on selecting Generate Table Compression Scripts, see To Generate Table Compression Scripts on page 2-22.

   The project components are generated. This might take a few minutes.

> **Note:** NetBeans automatically saves your project at the completion a successful build.

To Generate Table Compression Scripts

The database compression helps to reduce the resources and the costs of managing large data volumes. You can optionally create the database scripts with table compression enabled. Before you do so, make sure your database license includes the Oracle Advanced compression options.

- OHMPI compresses all the database tables based on the definition in object.xml of the MPI project in addition to the OHMPI specific tables.

- OHMPI provides Advanced compression for Oracle Database 12c and OLTP compression for Oracle Database 11g.

To enable the table compression:

1. Right-click the Master Person Index or the IHE Profile project in the Projects pane and then select **Generate Table Compression Scripts** to generate create.sql script with table compression or blob deduplication for Oracle Database 11g or 12c.

    The Choose Required Options dialog box appears.

2. Select the appropriate option for the Oracle database version.

3. Select the **BLOB Deduplication** check box to enable BLOB deduplication.

4. Click **Generate Files**.

For more information, see *Oracle Database Administrator's Guide* and *Oracle Database SecureFiles and Large Objects Developer's Guide*.

To Update a Project

Projects which are created from the earlier OHMPI releases include some outdated schema or library files that do not have the latest and additional features. When you update such projects, the Update function lists all the configuration schema and library files that you must update.

1. Right-click the Master Person Index or the IHE Profile project in the Projects pane and then select **Update**.

    The Update MPI Project dialog box appears listing all outdated files. By default, check boxes for all files are selected. You can unselect the **Will be updated** check box for the files which you do not want to update.

    > **Note:** You must back up the original files before updating the project.

2. Click **OK**.

To Regenerate the Project

1. Save any configuration changes to the Master Person Index or the IHE Profile project.

2. If you are using the command-line reports, back them up to a different directory. Generating the application overwrites the existing report client.

3. Right-click the Master Person Index or the IHE Profile project in the Projects pane and then select **Clean and Build**.

    The project files are cleaned and built. This might take a few minutes.

4. Right-click the Master Person Index or the IHE Profile project in the Projects pane and then select **Generate Master Index Files** or **Generate Table Compression Scripts**. For more information on selecting Generate Table Compression Scripts, see To Generate Table Compression Scripts on page 2-22.

The application components are regenerated. This might take a few minutes. The output panel displays the status and any errors that occur.

> **Note:** NetBeans automatically saves your project at the completion a successful build.

Upgrading 3.x Master Index Application to 4.0 Release

The wizard provides a simple procedure for you to upgrade any existing 3.x master index application which is created by the 3.x OHMPI release to the 4.0 release. The upgrade does not impact any existing data model schema, settings and configurations and database. This section provides instructions for upgrading any existing 3.x project using the wizard. To upgrade the existing 3.x project, perform the following steps:

1. Open the project.

> **Note:** Do not modify any existing data model, settings, and databases.

  a. On the NetBeans toolbar, click **Open Project**.

  b. Browse to the folder where the project is contained.

  c. Select the **Open Required Projects** check box and select **Open Project**.

  When project opens, the "Resolve project problems" warning appears.

  d. Click **Resolve Problems** and follow instructions to include the *junit* library.

2. Clean up the project artifacts that need to be regenerated by the new wizard.

  a. Right-click on the project in the NetBeans projects explorer, and select **Clean**.

3. Update the project. For information, see To Update a Project on page 2-22.

> **Note:** You must update the following files:
>
> - BusinessName/lib/standardizer-sbme-impl.jar
>
> - BusinessName/instance/Generic/lib/standardizer-sbme-generic-businessname.jar
>
> - Address/lib/standardizer-sbme-impl.jar

4. Regenerate the project artifacts using the new wizard.

  a. Right-click on the project in the NetBeans projects explorer, and select **Generate Master Index Files** or **Generate Table Compression Scripts**. For more information on selecting Generate Table Compression Scripts, see To Generate Table Compression Scripts on page 2-22.

5. Rebuild the project in order to automatically include all the release 4.0 artifacts.

  a. Right-click on the project in the NetBeans projects explorer, and select **Build**.

6. Deploy the upgraded application to the WebLogic Server 12c.

> **Note:** You can use the existing database which runs for the existing 3.x application.

a. Configure the application server.

b. Configure connectivity.

c. Deploy the build.

For any existing 3.x profiler, cleanser and loader, you must follow the instructions to regenerate the profiler, cleanser, and loader using the release 4.0 wizard. For information, see *Oracle Healthcare Master Person Index Loading the Initial Data Set User's Guide*. No changes on the existing configuration files are required to work with the upgraded profiler, cleanser, and loader.

3

Oracle Healthcare Master Person Index Databases

This chapter presents general information that will help you analyze your database requirements and what you should consider before you design an Oracle Healthcare Master Person Index database. It then provides a procedure that leads you through the process of creating an OHMPI database.

> **Note:** This is chapter is intended for users who need to set up a database for a project created using Oracle Healthcare Master Person Index.

This chapter includes the following sections:

- Understanding Oracle Healthcare Master Person Index Database Scripts and Design on page 3-1
- Creating an Oracle Healthcare Master Person Index Database on page 3-9

Understanding Oracle Healthcare Master Person Index Database Scripts and Design

Before you create an OHMPI database, familiarize yourself with the database scripts and the database structure. Analyze your database requirements, including hardware considerations, startup data, indexing needs, and performance.

The following sections provide information to help you in your analysis:

- Oracle Healthcare Master Person Index Database Scripts on page 3-2
- Oracle Healthcare Master Person Index Database Requirements on page 3-2
- Oracle Healthcare Master Person Index Database Structure on page 3-3
- Designing the Oracle Healthcare Master Person Index Database on page 3-3
- Oracle Healthcare Master Person Index Database Tablespaces on page 3-6
- Oracle Healthcare Master Person Index Database Table Description for sbyn_ systems on page 3-8
- Oracle Healthcare Master Person Index Database Table Description for sbyn_user_ code on page 3-9

Oracle Healthcare Master Person Index Database Scripts

The wizard creates SQL scripts based on information you specified about code lists and external systems that you can use to define startup data for the Master Person Index application. When you generate the application, additional scripts are generated for creating or dropping database tables, creating and dropping tables' indexes. These scripts appear under the Database Script node of the Master Person Index project, and are named `create.sql`, `systems.sql`, `codelist.sql`, `drop.sql`, `create-index.sql`, and `drop-index.sql`. You can modify these scripts as needed to customize the tables, indexes, startup data, and database distribution. You can also create new database scripts if needed.

Oracle Healthcare Master Person Index Database Requirements

When configuring the master person index database, there are several factors to consider, including basic software requirements, operating systems, and disk space. This section provides a summary of requirements for the database. For more detailed information about designing and implementing the database, refer to the appropriate database platform documentation. The person responsible for the database configuration should be a database administrator familiar with the OHMPI database and with your data processing requirements.

Database Platform Requirements

The OHMPI database can be run on Oracle12c R1 (12.1.0.2.0 or higher) or Oracle 11gR2. You must have this software installed before beginning the database installation. Make sure you also install the latest patches for the version you are using.

Securing Your Application Requirements

The OHMPI application can be secured via Application Server provided SSL. The OHMPI application runs as an EJB application and as a WSDL-based web service. Please refer to the Oracle WebLogic Server security guides on how to configure applications using SSL.

Operating System Requirements

The database can be installed on any operating system supported by the database platform you are using. See the documentation that came with your database server for more information.

Hardware Requirements

This section describes the minimum recommended hardware configuration for a database installation. These requirements are based on the minimum requirements recommended by Oracle database for a typical installation. Depending on the size of the database and expected volume, you should increase these recommendations as needed. See the documentation for your database for more information and for supported operating systems.

Oracle Database

For a Windows database server, the following configuration is recommended as a minimal installation:

- Windows 2008, Windows 2008 R2 on 64 bits, or Windows 7
- Pentium 266 or later

- 1 GB RAM (increase this based on the number of users, connections to the database, and volume)

- Virtual memory should be double the amount of RAM

- 3 GB disk space plus an additional 2 KB for each system record to be stored in the database (note that this is a conservative estimate per system record, assuming that most records do not contain complete data). This depends on the Oracle environment you install. Enterprise Edition can take up to 5 GB.

- 256-color video

For a UNIX database server, the following configuration is recommended as a minimal installation:

- 256 MB RAM (increase this based on the number of users and connections to the database)

- Swap space should be a minimum of twice the amount of RAM

- 2 GB disk space plus an additional 2 KB for each system record to be stored in the database (note that this is a conservative estimate per system record, assuming that most records do not contain complete data).

> **Note:** Disk space recommendations do not take into account the volume and processing requirements or the number of users. These are minimal requirements to install a generic database. At a minimum, the empty database and the database software will require 2.5 GB of disk space.

Oracle Healthcare Master Person Index Database Structure

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

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

For a complete description of the database tables, see *Oracle Healthcare Master Person Index Message Processing Reference*.

Designing the Oracle Healthcare Master Person Index Database

In designing the database, there are several factors to consider, such as the volume of data stored in the database and the number of transactions processed by the database daily. The OHMPI database should be created in its own tablespaces. The following sections describe some of the analyses to perform along with considerations to take into account when designing a database.

Designing for Performance Optimization

The Oracle database installation guide provides detailed information about installing the database software for optimal performance. The database platform includes guide containing information about monitoring and fine-tuning your database, including tuning memory, swap space, I/O, CPU usage, block and file size. You should be familiar with these concepts prior to creating a database.

Data Structure Analysis

Before defining the object structure, analyze the structure of the legacy data to help you define the object structure and the attributes of each field. You can use this data analysis to determine the amount of data that will be stored in the database, which will help you size the OHMPI database and decide how to best distribute the database. Knowing the volume of existing data plus the expected daily transaction volume will help you plan the requirements of the database server, such as networking needs, disk space, memory, and swap space.

The data structure analysis also helps you determine the processing codes and descriptions to enter in the common tables (described below), and should help you determine any default values that have been entered into certain fields that could skew the matching probability weights.

Common Table Data

Common table data analysis involves gathering information about the abbreviations used for specific data elements in each sending system, such as system codes and codes for certain attributes of the objects in your database. For example, if you are indexing person objects, there might be processing codes for genders, such as **F** for female or **M** for male. The processing codes and their descriptions are stored in a set of database tables known as common maintenance tables. The wizard creates a script to help you load the processing codes into the database.

When an enterprise object appears on the MIDM, the master person index application translates the processing codes defined in the common tables into their descriptions so the user is not required to decipher each code. The data elements stored in the common maintenance tables are also used to populate the drop-down lists that appear for certain fields in the MIDM. Users can select from these options to populate the associated fields.

User Code Data

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

Database Considerations

When you create the master person index database, you need to consider several factors, such as sizing, distribution, indexes, and extents. By default, all of the OHMPI database tables for an Oracle database are installed in the system tablespace. You should install the OHMPI tables in different tablespaces, depending on the original size and expected volume of the database.

Database Sizing

To begin the database installation, you first create a database instance using the provided configuration tools or command line functions. Use the tools provided by the database vendor to define the tablespace and extent sizing for the database.

Database Distribution

When you create the database instance, you can define the distribution of your system tables, data tables, rollback logs, dump files, and control files. Use internal policies regarding relational database distribution to determine how to best distribute your OHMPI database.

Database Indexes

The database index improves the performance of data retrieval and search database operations generically. By default, indexes are defined for the following system common tables:

- sbyn_appl

- sbyn_common_header

- sbyn_common_detail

- sbyn_enterprise

- sbyn_transaction

- sbyn_assumedmatch

- sbyn_potentialduplicates

- sbyn_audit

- sbyn_merge

Also, composite indexes are defined automatically for all the combined fields used for blocking. For each query block defined in the blocking query, a composite index containing the fields in that block is created automatically. If a field in that block is standard field or phonetic-coded field, an index for its original field is also created. Indexes are optionally defined for all the fields used for matching. You can create additional indexes against the database to optimize the searching and matching processes.

The following indexes are automatically created to improve performance when running large reports from the command line or MIDM:

CREATE INDEX SBYN_POTENTIALDUPLICATES3 ON SBYN_POTENTIALDUPLICATES (TRANSACTIONNUMBER ASC);

CREATE INDEX SBYN_ASSUMEDMATCH2 ON SBYN_ASSUMEDMATCH (TRANSACTIONNUMBER ASC);

CREATE INDEX SBYN_TRANSACTION4 on SBYN_TRANSACTION (EUID2 ASC, TIMESTAMP ASC);

CREATE INDEX SBYN_TRANSACTION3 on SBYN_TRANSACTION (TIMESTAMP ASC, TRANSACTIONNUMBER ASC);

> **Note:** To improve performance, these four indexes should be dropped prior to performing an initial load or batch load of data. They can be recreated once the load is complete if you are running the provided reports.

Oracle Healthcare Master Person Index Database Tablespaces

The OHMPI automatically creates four database tablespaces, assigns the OHMPI tables, and indexes to different tablespaces. Table 3–1 lists the database tablespaces created by OHMPI.

Table 3–1   Database Tablespaces

Tablespace Name	Description
<application_name>.dbf	Application tablespace for all the OHMPI tables except the SBYN_TRANSACTION table
<application_name>_transaction.dbf	Application transaction tablespace for the SBYN_TRANSACTION table
<application_name>_index.dbf	Application indexes tablespace for OHMPI table indexes
<application_name>_temp.dbf	Application temporary tablespace for carrying out temporary database transactions, used by database system when required

When you create an OHMPI application through the NetBeans OHMPI wizard, tablespaces are created and assigned automatically.

The commands for creating and assigning tablespaces are generated along with tables and indexes in the following scripts under the <project_home>/src/DatabaseScript directory of an OHMPI project:

- create.sql

- create-index.sql

By default, the tablespace is locale datafile storage with the maximum size of 100 MB. The default tablespaces are created at location specified by the Oracle database property DB_CREATE_FILE_DEST.

For example,

```
CREATE TABLESPACE PERSON40
DATAFILE 'PERSON40.DBF'
SIZE 100M REUSE
AUTOEXTEND ON;


CREATE TABLESPACE PERSON40_TRANSACTION
DATAFILE 'PERSON40_TRANSACTION.DBF'
SIZE 100M REUSE
AUTOEXTEND ON;


CREATE TEMPORARY TABLESPACE PERSON40_TEMP
TEMPFILE 'PERSON40_TEMP.DBF'
SIZE 100M REUSE
AUTOEXTEND ON;


CREATE TABLESPACE PERSON40_INDEX
DATAFILE 'PERSON40_INDEX.DBF'
SIZE 100M REUSE
AUTOEXTEND ON;
```

When you drop OHMPI tables and indexes using the following scripts in the Database Scripts folder of an OHMPI project, the corresponding tablespaces are also dropped:

- drop.sql

- drop-index.sql

Managing Oracle Healthcare Master Person Index Database Tablespaces

OHMPI automatically creates the default tablespace with the following settings when a tablespace is created:

- SMALLFILE tablespace

- Datafile is created at $ORACLE_HOME

- Tablespace size is 100 MB and is auto-extendable

You can customize these settings according to the actual requirements and platforms. You must modify the create.sql script directly using any text editor.

Creating a BIGFILE Tablespace

Specify the BIGFILE keyword to modify the table creation statement. For example:

```
CREATE BIGFILE TABLESPACE PERSON40
DATAFILE 'PERSON40.DBF'
SIZE 100M REUSE
AUTOEXTEND ON;
```

Storing the DATAFILE on Different Storage Location

Specify different DATAFILE path to modify the tablespace creation statement. For example:

- For Linux:

```
CREATE TABLESPACE PERSON40
DATAFILE '/users/tablespace/PERSON40.DBF'
SIZE 100M REUSE
AUTOEXTEND ON;
```
- For Windows:

```
CREATE TABLESPACE PERSON40
DATAFILE 'C:\users\tablespace\PERSON40.DBF'
SIZE 100M REUSE
AUTOEXTEND ON;
```

Increasing Default Initial Tablespace Size

Change the default initial size of the tablespace to modify the tablespace creation statement. For example:

```
CREATE TABLESPACE PERSON40
DATAFILE 'PERSON40.DBF'
SIZE 200M REUSE
AUTOEXTEND ON;
```
For more advanced guidelines on how to manage and tune database tablespaces, see *Oracle Database Administrator's Guide for 12c Release*.

Oracle Healthcare Master Person Index Database Table Description for sbyn_systems

The following table lists and describes the columns in the sbyn_systems database table so you can create SQL statements to insert source system information into the master person index database.

Column	Description
systemcode	The unique processing code of the system. This field accepts any of the following characters:
	■ ! _ ~ ( ) { } + \Q # $ % & : ; - /
	■ a-z
	■ A-Z
	■ 0-9
	■ þ ÿ Þ ß 'à-ö ø-ý À-Ö Ø-Ý
description	A brief description of the system, or the system name.
status	The status of the system in the master person index system. Specify **A** for active or **D** for deactivated.
id_length	The length of the local identifiers assigned by the system. This length does not include any additional characters added by the input mask (see below).
	Note: The default maximum length of the LID database columns is 25. If the system generates longer local IDs, be sure to increase the length of all LID columns in the database.
format	The required data pattern for the local IDs assigned by the system. For more information about possible values and using Java patterns, see "Patterns" in the class list for `java.util.regex` in the Javadocs provided with Java. Note that the pattern specified here might be further restricted by the input mask described below.
input_mask	A mask used by the MIDM to add punctuation to the local ID. For example, you can add an input mask to display the local IDs with hyphens or constant characters. To define an input mask, enter a character type for each character in the field and place any necessary punctuation between the types. For example, to insert a hyphen after the second and fifth characters in an 8-digit ID, the input mask would be **DD-DDD-DDD**. The following character types can be used; any other characters are treated as constants.
	■ **D** - indicates a numeric character.
	■ **L** - indicates an alphabetic character.
	■ **A** - indicates an alphanumeric character.
	If you use an input mask, you should also define a value mask to remove the punctuation from the stored value.
value_mask	A mask used to strip any extra characters that were added by the input mask. This mask ensures that data is stored in the database in the correct format and matches the regex format for validation.
	To specify a value mask, type the same value entered for the input mask, but type an "x" in place of each punctuation mark. Using the 8-digit input mask described above, you would specify a value mask of **DDxDDDxDDD**. This strips the hyphens before storing the ID in the database.
create_date	The date system information has inserted into the database. You can specify "sysdate" for this column, or use the variables defined in of the sample script.

Column	Description
create_userid	The logon ID of the user who inserted the system information into the database. You can enter the logon ID or use the variables defined in of the sample script.

Oracle Healthcare Master Person Index Database Table Description for sbyn_user_code

The following table lists and describes the columns in the sbyn_user_code database table so you can create SQL statements to insert user code data into the master person index database.

Column Name	Description
code_list	The code list name of the user code type (using the credit card example above, this might be similar to "CREDCARD"). This column links the values for each list.
code	The processing code of each user code element.
description	A brief description or name for the user code element. This is the value that appears in the drop-down list.
format	The required data pattern for the field that is constrained by the user code. For more information about possible values and using Java patterns, see "Patterns" in the class list for java.util.regex in the Javadocs provided with Java. Note that the pattern might be further restricted by the value of the input mask described below.
input-mask	A mask used by the MIDM to add punctuation to the constrained field. For example, the input mask **DD-DDD-DDD** inserts a hyphen after the second and fifth characters in an 8-digit ID. If you use an input mask, you should also use a value mask to strip the punctuation for database storage (see below). The following character types can be used. ■ **D** - Numeric character ■ **L** - Alphabetic character ■ **A** - Alphanumeric character
value-mask	A mask used to strip any extra characters that were added by the input mask for database storage. The value mask is the same as the input mask, but with an "x" in place of each punctuation mark. Using the input mask described above, the value mask is **DDxDDDxDDD**. This strips the hyphens before storing the ID.

Creating an Oracle Healthcare Master Person Index Database

Once you have customized the configuration files and generated the Master Person Index application, you can create the OHMPI database. Before you begin, make sure you have Oracle installed on the database server.

During this process you can define custom startup data, such as code lists and source systems. The wizard defines SQL statements in the Code List and Systems SQL files (codelist.sql and system.sql) to insert startup data into the database based on information you specify, and you can customize the statements to insert the data relevant to your object structure. The code lists you define are used to translate processing codes from incoming messages into descriptions for the MIDM fields and to create drop-down lists for MIDM fields. System information is required in order to add records to the OHMPI application.

Follow these steps to create the database:

You can also delete the database for testing purposes using the supplied script. See Dropping Oracle Healthcare Master Person Index Database Tables on page 4-6 for more information.

Step 1: Analyze the OHMPI Database Requirements

Before you begin to create the master person index database, perform an analysis of the structure of the legacy data to be stored in the database and determine the amount of data that will be processed daily. During the analysis, be sure to define the processing codes that need to be stored in the common maintenance tables and the systems that will share data with the master person index application. You should also know the length and format of the local IDs assigned by each system.

A database administrator who is familiar with your data and processing requirements should perform this task. After this task is complete, continue to Step 2: Create a Master Person Index Database and User.

For additional information and guidelines about how to set up your database, see Understanding Oracle Healthcare Master Person Index Database Scripts and Design on page 3-1.

Step 2: Create a Master Person Index Database and User

Before beginning this step, complete Step 1: Analyze the OHMPI Database Requirements. After you create the database instance and user, continue to Step 3: Define Master Person Index Database Indexes if you want to define additional database indexes; otherwise skip to Step 4: Define Master Person Index External Systems.

For this step you need to create a database in which the master person index database instance will be created. Use the tools provided by your database vendor to create the database. Using these tools, you define tablespaces, including their sizes and locations; extents; and dump file, log file, and rollback file sizes and locations. Make sure these issues have been thoroughly analyzed and designed before creating the database.

Once you create the database, you can use standard SQL to create the master person index application user for the database. The user you create in this step will be used to create the database structure and to connect to the database through the application server.

For Oracle, assign the user to the "connect" and "resource" roles for the master person index tablespaces. For example:

```
create user <ohmpi_database_username> identified by <ohmpi_database_password>;
grant connect, resource to <ohmpi_database_username>;
```

```
grant create tablespace to <ohmpi_database_username>;
grant drop tablespace to <ohmpi_database_username>;
commit;
```

where username is the login ID of the administrator user and password is the login password of the administrator user.

Step 3: Define Master Person Index Database Indexes

To optimize data processing in the Master Person Index application, you can define additional indexes for the database tables that store object data. Oracle recommends that you define indexes for each field used for searching, blocking, or matching. You can define these indexes in create-index.sql or create.sql script. Oracle recommends that you use create-index.sql for any domain-relevant indexes and create.sql for any system-relevant common indexes. Before you begin this step, complete Step 2: Create a Master Person Index Database and User.

To Define an Index

1. Perform one of the following:

    ■   Under the master person index project in the Projects window, expand Database Script and then open **create-index.sql** or **create.sql** in the text editor.

    ■   Navigate to project_home/src/DatabaseScript, where project_home is the location of the master person index project files. Open **create-index.sql** or **create.sql** in a text editor.

2. Perform one of the following:

    ■   Remove an existing index definition (not recommended).

    ■   Create new index definitions for the required fields.

    ■   Modify an existing index definition.

3. Save and close the file.

4. Continue to Step 4: Define Master Person Index External Systems.

Step 4: Define Master Person Index External Systems

A SQL script is automatically created to insert the systems you specified in the wizard. These statements are provided in systems.sql in the master person index project. Before you begin this step, complete Step 2: Create a Master Person Index Database and User and optionally, Step 3: Define Master Person Index Database Indexes.

To Define an External System

1. Perform one of the following:

    ■   Under the master person index project in the Projects window, expand Database Script and then open **systems.sql** in the text editor.

    ■   On your computer, navigate to *project_home*/src/DatabaseScript, where *project_home* is the location of the master person index project files. Open **systems.sql** in a text editor.

2. For each system, create an INSERT statement using the column descriptions in Master Person Index Database Table Description for sbyn_systems to define the VALUES clause.

For example:

```
INSERT into sbyn_systems (systemcode, description, status, id_length,
format, input_mask, value_mask, create_date, create_userid) VALUES ('ARS',
'Automated Registration System', 'A', 8, '[0-9]{8}', 'DD-DDD-DDD',
'DD^DDD^DDD', sysdate, 'admin');
```

3. Delete any default INSERT statements you do not need.

4. Save and close the file.

5. Continue to Step 5: Define Master Person Index Code Lists.

Step 5: Define Master Person Index Code Lists

You only need to perform this step if you defined any fields in the object structure to have a code module. The SQL script for entering processing codes and descriptions into the database is written in PL/SQL. The wizard creates a stanza in codelist.sql (located under the Database Script node of the project) for each code list you specified in the field properties. You need to customize the file by defining the entries for each code list. This script inserts data into two tables: sbyn_common_header, which lists the types of common table data, and sbyn_common_detail, which lists each common table data element. Before you begin this step, complete Step 4: Define Master Person Index External Systems.

> **Note:** The codes you specify in this file can be no longer than eight characters (the codes are the second value in the value list for each common table data type and data element).

For information on the syntax, see To Customize Common Table Data for Oracle on page 3-12.

To Customize Common Table Data for Oracle

1. Perform one of the following:

   - Under the master person index project in the Projects pane, expand Database Scripts and then open **codelist.sql** in the text editor.

   - On your computer, navigate to *project_home*/src/DatabaseScript, where *project_home* is the location of the master person index project files. Open **codelist.sql** in a text editor.

2. Scroll to the following line.

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

   The statements following this line must be customized.

3. In the first code list stanza, change "module description" in the first line to a brief description of the code type.

   For example:

   ```
   -- ****       PHONTYPE   ****
   tCode('L', 'PHONTYPE', 'TELEPHONE TYPE'),
   ```

4. Create the entries for the module using the following syntax:

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

where "code" is the processing code of the data element and "code description" is the description of the element as you want it to appear on the Master Index Data Manager windows. For example:

```
-- ****          PHONTYPE   ****
tCode('L', 'PHONTYPE', 'TELEPHONE TYPE'),
tCode('V', 'H', 'HOME'),
tCode('V', 'C', 'CELL'),
tCode('V', 'F', 'FAX'),
tCode('V', 'O', 'OFFICE'),
tCode('V', 'HB', 'HOME BUSINESS'),
```

5. Repeat the previous two steps for each code list type defined in the file.

6. If you specified additional code list fields in `object.xml` and `midm.xml` after the database scripts were generated, add a new stanza for each new code type.

7. In the last code module stanza, make sure each line except the last contains a comma at the end.

   For example:

```
-- ****          ADDRTYPE   ****
tCode('L', 'ADDRTYPE', 'ADDRESS TYPE'),
tCode('V', 'H', 'HOME'),
tCode('V', 'B', 'BUSINESS'),
tCode('V', 'M', 'MAILING')
```

8. Save and close the file.

9. Perform one of the following:

   - If you need to define user code lists, continue to Step 6: Define Master Person Index User Code Lists.

   - If you need to create a custom database script, skip to Step 7: Create Custom Master Person Index Database Scripts.

   - If you are ready to create the master person index database structure, skip to Step 8: Create the Master Person Index Database Structure.

Step 6: Define Master Person Index User Code Lists

If you specified a value for the Constrained By and User Code properties of a field, you must define the user code values for those fields. Below is a sample insert statement for the sbyn_user_code table.

```
insert into sbyn_user_code (code_list, code, descr, format, input_mask,
value_mask) values ('AUXIDDEF', 'CC', 'CREDIT CARD', '[0-9]{16}',
'DDDD-DDDD-DDDD-DDDD',
'DDDD^DDDD^DDDD^DDDD');
```

To learn more about the sbyn_user_code table, see Oracle Healthcare Master Person Index Database Table Description for sbyn_user_code on page 3-9. Complete Step 5: Define Master Person Index Code Lists before beginning this step.

To Define a User Code List

1. Create a new SQL file to contain the user code SQL statements.

2. Use the above sample to define a value for the user code drop-down list and the required format for the dependent fields.

3. Repeat the above step for each drop-down list value and type (for example you might have one list for credit cards and another for postal codes and their corresponding cities).

4. Save and close the file.

5. Continue to Step 7: Create Custom Master Person Index Database Scripts or to Step 8: Create the Master Person Index Database Structure if you do not need to create any custom database scripts.

Step 7: Create Custom Master Person Index Database Scripts

You can insert additional information into the database by creating a custom script. For information about the structure of the OHMPI database, see *Oracle Healthcare Master Person Index Message Processing Reference*.

To Create a Custom Script

1. Create a new text file to contain the SQL statements.

2. In the text editor, create the SQL script to insert the custom data.

3. Save and close the file.

4. Continue to Step 8: Create the Master Person Index Database Structure.

Step 8: Create the Master Person Index Database Structure

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

To Create the Database Structure

1. Open a standard SQL editor or SQL command line and connect to the master person index database using the user you created in Step 2: Create a Master Person Index Database and User.

2. Perform one of the following:

   > **Note:** To customize the tablespaces, modify the create.sql script. For information, see Oracle Healthcare Master Person Index Database Tablespaces on page 3-6.

   - Open **create.sql** in the NetBeans editor, copy and paste the entire text of the file into the SQL editor, and run the script against the database.

   - From the SQL editor or command line, run **create.sql** directly. The file is located at *project_home*/src/DatabaseScript.

     > **Tip:** To run this script for Oracle, the command is @create.sql.

3. Repeat the previous two steps for **systems.sql**, **codelist.sql**, **create-index.sql**, and any custom scripts you created.

Step 9: Specify a Starting EUID for a Master Person Index

By default, the EUIDs assigned by the master person index application start with "0", with padded zeroes added to the left to make the EUID number the correct length. For

more information, see *Oracle Healthcare Master Person Index Configuration Reference*. You can modify this numbering format by changing the value of the seq_name column of the sbyn_seq_table database table where the sequence name is "EUID". For example:

```
update sbyn_seq_table set seq_count=1000000001 where seq_name='EUID';
```

4

Database Connection Pools, Data Resources, Deleting Tables, and Configuring JMS

This chapter provides procedures on how to define database connection pools and data resources, which each master person index application requires, and deleting database tables that are no longer needed.

This chapter includes the following sections:

Defining the Database Connection Pools and Data Resources

Each master person index application requires two database connection pools; one for the master controller and one for the sequence manager. To set up the connection pools, you need to create the connection pools and then define a JDBC resource for each. This section provides general instructions for setting up the connection pools. For more information about the procedures in this section, see the online help provided with your server's Admin Console.

> **Note:** In the past if you used an Oracle database, you needed to manually install or copy the database driver to your application server environment. This is no longer the case as the drivers are now included with the OHMPI installation.

To Create JDBC Connection Pools and Data Resources

The JDBC connection pools provide connections to the master person index database. You need to create two connection pools that are configured in the same way.

A JDBC resource (also known as a data source) gives the master person index application the ability to connect to the database. Two JDBC resources are required.

Before proceeding, make sure you have the relevant information about the master person index database (such as the database name, URL, and administrator login credentials).

Define database connectivity through the application server. See Creating JDBC Connection Pools and Setting Data Resources on page 4-2.

Creating JDBC Connection Pools and Setting Data Resources

To Create JDBC Connection Pools and Data Resources for an MPI Application Project for Oracle

1. Start and stop the Oracle WebLogic Server. For instructions, see *Oracle Fusion Middleware Online Documentation 12c Release 1 (12.1.3)*.

2. Launch the **Oracle WebLogic Server Administration Console**.

3. Log in using the default user name (**weblogic**) and password (**welcome1**).

   The Oracle WebLogic Administration Console appears.

4. On the left panel, under Domain Structure, expand **Services**, click **JDBC**, and then select **Data Sources**.

   A Summary of JDBC Data Sources appears in the right panel.

5. To create a new JDBC Data Source click **New** at the bottom of the right panel.

   Settings for a new JDBC Data Source appears in the right panel of the page. It is here that you will create a new JDBC Data Source.

6. In the **Name** field, type *object_name*DataSource.

   The name you enter here will propagate elsewhere, and it must be the object name that you have already created (for example, PersonDataSource).

7. In the **JNDI Name** field, type jdbc/*object_name*DataSource.

   Use the name you entered in step 6 here (for example, **jdbc/PersonDataSource**).

8. Click **Save**.

   A new page appears in the right panel which is for setting the Database Type.

9. In the **Database Type** drop-down list, select the appropriate type (for example, **Oracle**).

10. In the **Database Driver** drop-down list, select the appropriate driver. For example, **Oracle's Driver (Thin XA) for Instance Connections; Versions: 9.0.1; 9.2.0; 10, 11**.

11. Click **Next**.

12. Click **Next**.

    Connection Properties appears on the Create a New JDBC Data Source panel. Use it to define the connection properties.

13. In the **Database Name** field, type a name for the database to which you want to connect (for example: Person).

14. In the **Host Name** field, type the name or the IP address of the database server (for example: localhost).

15. In the **Port** field, type the port on the database server that is used to connect to the database (for example: 1521, the default for Oracle).

16. In the **Database User Name** field, type the database account user name you want to use to create database connections (for example: root).

17. In the **Password** field, type a password for your database account to use to create database connections.

18. In the **Confirm Password** field, re-type the password to confirm it.

19. Click **Next**.

The Settings for *object_name*DataSource page appears in the right panel.

20. Click the **Connection Pool** tab, click **Test Configuration**, and then click **Next**.

Select Targets appears on the Create a New JDBC Data Source page in the right panel. Here you select one or more targets to deploy the new JDBC data source.

21. In the Servers check list, select one or more target servers and click **Finish**.

> **Note:** If you do not select a target, the data source will be created but not deployed. You will need to deploy the data source at a later time.

22. Repeat the above steps to create jdbc/*object_name*SequenceDataSource.

You are now ready to deploy your application.

Creating JMS Resources for an MPI Application Project

The OHMPI delivers the outbound notification events to an OHMPI application-specific JMS topic on any update of the operations for the enterprise objects. You can subscribe the OHMPI application specific JMS topic to receive the outbound notification events for processing.

The following are the types of JMS subscribers:

- **Non-durable Subscriber**: A non-durable subscriber receives only events that are subscribed while it is active.

- **Durable Subscriber**: For a durable subscriber, the subscribed events are retained in the JMS provider even if the durable subscriber is inactive until the events are received after the durable subscriber is active or the events expire. Therefore, the events can be accumulated in the unexpected growth while the durable subscriber is inactive.

You must choose a durable subscriber or a non-durable subscriber based on the business use cases and handle it reasonably. This section describes how to configure the JMS resources for the MPI application.

JMS servers act as management containers for the queues and topics in the JMS modules that are targeted to them.

The following procedure includes instructions for creating JMS resources, which includes a:

- JMS Server
- JMS Module
- JMS Connection Factory in the specific JMS Module
- JMS Topic in the specific JMS Module

To Create JMS Server

1. On the left panel, under Domain Structure, expand **Services**, click **Messaging**, and then select **JMS Servers**.

A Summary of JMS Servers appears in the right panel. It includes a table that summarizes the JMS servers that have been created in the current WebLogic Server domain.

2. In the table of previously created JMS Servers, click **New**.

   The Create a New JMS Server panel appears.

3. In the **Name** field, type the name for your new JMS Server.

   > **Note:** This name already exists in the table of previously created JMS Servers (in the example, **PersonJMSServer**).

4. Click **Next**.

   Select targets appears in the right panel under Create a New JMS Server.

5. From the **Target** drop-down list select a target server instance or migratable target on which you want to deploy the JMS Server.

   If you are deploying on a Clustered environment, select the appropriate target server with the cluster environment.

   > **Note:** The default server instance is exampleServer.

6. Click **Finish**.

To Create JMS Module

1. On the left panel, under Domain Structure, expand **Services**, click **Messaging**, and then select **JMS Modules**.

   The JMS Modules panel appears.

2. In the JMS Modules table, click **New** to add a new JMS Module.

   The Create JMS System Module panel appears.

3. In the **Name** field, type the new JMS Module name (for example, *mpi_applicationName*JMSModule).

   > **Note:** Again, remain consistent to the name chosen for the JDBC Data Source and the JMS Server (in the previous examples the key word was "Person," making this name **PersonJMSModule**).

4. Click **Next**.

   Targets appears in the right panel under Create a New JMS System Module.

5. In the Servers area, select the server or cluster on which you want to deploy this JMS system module.

   If you are deploying on a Clustered environment, select the appropriate target server with the cluster environment.

   > **Note:** Retain the default, examplesServer.

6. Click **Finish**.

To Create JMS Connection Factory

1. On the left panel, under Domain Structure, expand **Services**, click **Messaging**, and then select **JMS Modules**.

2. Select the JMS Module (in the example, **PersonJMSModule**) from the table of JMS Modules.

   The Settings for PersonJMSModule page appears in the right panel.

3. In the Summary of Resources table, click **New**.

4. Under the Type column in the Summary of Resources table, select **Connection Factory**, and click **Next**.

   Another panel of Create a New JMS System Module Resource appears.

5. In the **Name** field, type `PersonOutBoundSender`.

6. In the **JNDI Name** field, type jms/*mpi_application_name*OutBoundSender.

7. Make sure that the **XA Connection Factory Enabled** check box is selected.

8. Click **Next**

9. In the **Target** field, retain the default server instance, which is exampleServer, and click **Finish**.

   If you are deploying on a Clustered environment, select the appropriate target server with the cluster environment.

To Create JMS Topic

1. On the left panel, under Domain Structure, expand **Services**, click **Messaging**, and then select **JMS Modules**.

2. In the right panel, select the JMS Module you created (in the example, **PersonJMSModule**) from the table of JMS Modules.

   Settings for PersonJMSModule appears in the right panel with a Summary of Resources table.

3. In the Summary of Resources table, click **New**, select **Topic**, and then click **Next**.

   The Create a New JMS System Module Resource panel appears on the right side of the window. Use this panel to set the properties that identify the new topic.

4. In the **Name** field, under JMS Destination Properties, type *mpi_application_name*Topic (for example, `PersonTopic`).

5. Set **jms/PersonTopic** as the JNDI Name and click **Next**.

   The Create a New JMS System Module Resource page appears in the right panel. Use this page to set the properties that will be used to target your new JMS system module resource.

6. In the **Subdeployments** drop-down list, select the topic name you just created (for example, `PersonTopic`) and click **Create a New Subdeployment**.

7. In the **Subdeployment Name** field, type *mpi_application_name*Topic (for example, `PersonTopic`), and click **OK**.

8. In the Targets table of JMS Servers, select *mpi_application_name***JMSServe**r (for example, **PersonJMSServer**).

   If you are deploying on a Clustered environment, select the appropriate target server with the cluster environment.

9. Click **Finish**.

Configuring JMS Resource

You can configure the JMS Resource, <APP-NAME>OutBoundSender, on the server deploying the EJB component. Configuring the JMS resource is optional.

Dropping Oracle Healthcare Master Person Index Database Tables

Scripts are provided to drop the default database tables and indexes created in Step 8: Create the Master Person Index Database Structure. This is useful while testing the master index application implementation.

To Delete Database Tables

1. Open a standard SQL editor and connect to the master person index database using the user you created in Step 2: Create a Master Person Index Database and User.

2. Open **drop.sql** in the NetBeans editor.

3. Copy and paste the entire text of drop.sql into the SQL editor.

4. Run the script against the database.

5. If the database is running on the Oracle 10**g** platform, open a SQL prompt and run the following command.

```
PURGE RECYCLEBIN;
```

5

Setting Up Application Servers to Use with Web Service

This chapter provides information you need to set in order to run a Master Person Index (MPI) web service on an application server.

This chapter includes the following section:

- Setting Up Servers with Users for the MPI Web Service on page 5-1

Setting Up Servers with Users for the MPI Web Service

The Master Person Index Application runs on the Oracle WebLogic Server. The following section provides the information you need to set up the server for Web Service.

- Creating User Accounts for Web Service on Application Server on page 5-1

Creating User Accounts for Web Service on Application Server

In the following steps, you create the `MasterIndex.Admin` group, and then create a new user:

1. On the left panel, under Domain Structure, expand **Services**, and then select **Security Realms**.

2. In the table on the Summary of Security Realms panel, click **myrealm**, which is the name of the realm.

    The Settings for myrealm panel appears.

3. Select the **Users and Groups** tab and then click **Groups**.

4. In the Groups table, click **New**.

5. In the **Name** field, type `MasterIndex.Admin` (if it does not exist) and click **OK**.

6. On the Settings for myrealm panel, select **Users and Groups** and then **Users**.

7. In the Users table, click **New**.

8. Type `MasterIndex.WSUser` and a password of your choosing for the new user you are creating and click **OK**.

9. Select **User Group**.

10. To add the user you created, drag **MasterIndex.Admin** from the **Available** list to the **Chosen** list.

6

Web Layer Separation for OHMPI

To separate the Web tier from the Business Logic tier, the MIDM UI layer is separated from the MIDM EJB layer. The MIDM UI layer and the MIDM EJB layer that are currently packaged into one EAR deployment package are broken down into individual WAR and EJB deployment packages.

Standalone WAR and EJB deployment packages are generated and these are setup on two different servers. The default EAR is also generated and is used for the consolidated deployment. The ohmpi-midm-connector.properties file is modified for setting up the WAR deployment to communicate with the EJB deployment.

To separate the web layer from the EJB layer:

1. Build the project. For information, see Building the Project on page 6-1.

2. Configure the application server. For information, see Configuring the Application Server on page 6-1.

3. Configure the connectivity. For information, see Configuring Connectivity on page 6-2.

4. Deploy the build. For information, see Deploying the Build on page 6-2.

Building the Project

- For a new project, two additional deployment packages (WAR and EJB) are available in the dist folder after you build the project. For example, -war.war and -ejb.jar.

- For existing projects, execute the Clean Project and Generate Master Index files to update the project before building it.

Configuring the Application Server

To configure the servers hosting the individual deployment packages:

1. Define users with the same username, password, and roles on both the servers.

2. Define the JDBC Data Sources (<APP-NAME>DataSource and <APP-NAME>SequenceDataSource) on the server deploying the EJB component.

3. Optional step. You can define the JMS Resource, <APP-NAME>OutBoundSender, on the server deploying the EJB component.

Configuring Connectivity

When you create a new project, the following command line utilities are generated in the *<project>* folder:

- `generate-wallet.bat`

- `generate-wallet.sh`

Generate the wallet contents using the `generate-wallet.bat` (for Microsoft Windows) and/or `generate-wallet.sh` (for UNIX) utility file.

1. Pass the following command line arguments:

   `<remote-weblogic-provider-url> <USERNAME> <PASSWORD> <Wallet Password>`
   - remote-weblogic-provider-url: For example, t3://<host>:<port>

   - USERNAME: Username defined during the user setup

   - PASSWORD: Password set during the user setup

   - Wallet Password: This must be a minimum of 8 character, alphanumeric, and have a special character.

   The *ohmpiWallet* folder is created in the project folder.

2. Copy the *ohmpiWallet* folder to the <app-server-domain-config-directory>/config folder in the web layer deployment server.

Deploying the Build

Deploy the EJB and WAR package on the EJB Server and Web Server, respectively.

7

Installing OHMPI on a WebLogic Cluster

You can deploy the Oracle Healthcare Master Person Index (OHMPI) application on a WebLogic cluster with preconfigured managed servers. The Person Template MPI application is used as an example here to describe the process of configuring the WebLogic server and deploying the Person MPI application to a WebLogic cluster. Deploy the complete MPI application to a single WebLogic cluster. These instructions are applicable to OHMPI 4.0 and WebLogic 12c.

For detailed information on WebLogic clustering feature and capabilities, see *Oracle Fusion Middleware Using Clusters for Oracle WebLogic Server Guide*.

This chapter contains the following sections:

Installing OHMPI Using OHMPI Installer

You can install OHMPI using the OHMPI installer by following the instructions in the *OHMPI Installation Guide*. Use the platform-specific installer appropriate to the operating system.

Creating and Configuring a WebLogic Machine and Node Manager

To create and configure WebLogic machine and node manager:

1. Log in to the WebLogic administration console.

2. In the left pane, click on your domain (for example, `domain1212`) and expand **Environment**.

3. Click **Machines** and then click **Lock & Edit**.

4. Click **New**.

The Create a New Machine screen appears.

5. Enter `ohmpi_machine` (or any other preferred name) in the **Name** field.

6. Click **Next**.

7. Specify the node manager properties.

   The node manager runs on localhost with SSL port 5556.

8. Click **Finish**.

9. Click **Activate Changes**.

Creating and Configuring the OHMPI Cluster

You can create and configure the OHMPI cluster in the following ways:

- Creating and Configuring a Standard Cluster
- Creating and Configuring a Dynamic Cluster

Creating and Configuring a Standard Cluster

To create and configure a standard cluster:

1. Log in to the WebLogic administration console.

2. In the left pane, click on your domain (for example, `domain1212`) and expand **Environment**.

3. Click **Clusters**.

4. Click **Lock & Edit**.

5. From the **New** drop-down list, select **Cluster**.

   The Create a New Cluster screen appears.

6. Enter cluster name in the **Name** field.

   For example, `ohmpi_cluster`.

7. From the **Messaging Mode** drop-down list, select **Unicast**.

8. Click **OK**.

9. In the Summary of Clusters pane, click **ohmpi_cluster**.

10. In the Settings for ohmpi_cluster screen, click the **Servers** tab.

11. Scroll down to the Servers table and click **Add**.

12. Enter server name in the **Server Name** field.

    For example, `ohmpi_server-1`.

13. Enter server listen address in the **Server Listen Address** field.

    For example, `localhost`.

14. Enter listen port for the first server in the **Server Listen Port** field.

    For example, `8001`.

15. Click **Finish**.

16. Click **ohmpi_server-1** to configure the server.

    The Settings for ohmpi_server-1 screen appears.

17. From the **Machine** drop-down list, select **ohmpi_machine**.

18. Select **Listen Port Enabled** and **SSL Listen Port Enabled** and enter the SSL listen port in the **SSL Listen Port** field.

    For example, `8101`.

19. Click **Save**.

20. Repeat steps 15 through 18 to specify settings for ohmpi_server-2.

    > **Note:** Specify 8002 as the Server Listen Port and 8102 as the SSL listen port for ohmpi_server-2.

21. Click **Activate Changes**.

Creating and Configuring a Dynamic Cluster

To create and configure a dynamic cluster:

1. Log in to the WebLogic administration console.

2. In the left pane, click on your domain (for example, `domain1212`) and expand **Environment**.

3. Click **Clusters**.

4. Click **Lock & Edit**.

5. From the **New** drop-down list, select **Dynamic Cluster**.

    The Create a New Dynamic Cluster screen appears.

6. Enter cluster name in the **Name** field.

    For example, `ohmpi_cluster`.

7. From the **Messaging Mode** drop-down list, select **Unicast**.

8. Click **Next**.

9. Enter `2` in the **Number of Dynamic Servers** field.

    > **Note:** Do not change the default values.

10. Click **Next**.

11. Select the required machine configuration. In this example, select **Use a single machine for all dynamic servers**.

12. Click **Next**.

13. Enter `8000` in the **Listen Port for First Server** field.

14. Enter `8100` in the **SSL Listen Port for First Server** field.

15. Click **Next**.

16. Review the configurations you have selected.

17. Click **Finish**.

18. Click **Activate Changes**.

Creating Application Server Resources to Target to the OHMPI Cluster

Creating Data Sources and Data Store

To create a JDBC data source:

1. Log in to the WebLogic administration console.

2. Click **Lock & Edit**.

3. In the left pane, click on your domain (for example, `domain1212`) and expand **Services**.

4. Select **Data Sources**.

> **Note:** For the Person MPI application, you must create the following data sources: <Application>DataSource and <Application>SequenceDataSource.
>
> For information on how to create the JDBC data sources for an MPI application project, see To Create JDBC Data Resources for an MPI Application Project for Oracle.

5. In the **Select Targets** section, select the **ohmpi_cluster** check box.

6. Select the **All servers in the cluster** option.

7. Click **Finish**.

To create a JMS data source:

1. In the Summary of JDBC Data Sources screen, select **Generic Data Source** from the **New** drop-down list.

   The Create a New JDBC Data Source screen appears.

2. Enter a data source name in the **Name** field.

   For example, `PersonJMSDataSource`.

3. Enter JNDI name in the **JNDI Name** field.

   For example, `jdbc/PersonJMSDataSource`.

4. Click **Next**.

5. Under JDBC Data Source Properties, select **Oracle's Driver (Thin) for Service connections; Versions: 9.0.1 and later** from the **Database Driver** drop-down list.

6. Click **Next**.

7. Deselect the **Supports Global Transactions** check box.

8. Click **Next**.

9. Enter values in the **Database Name**, **Host Name**, **Port**, **Database User Name**, and **Password** fields for the new JDBC data source.

10. Click **Next**.

11. Click **Test Configuration** to ensure that the connection is good.

12. Click **Next**.

13. In the **Select Targets** section, select the **ohmpi_cluster** check box.

14. Select the **All servers in the cluster** option.

15. Click **Finish**.

To create a data store:

1. Log in to the WebLogic administration console.

2. In the left pane, click on your domain (for example, domain1212) and expand **Services**.

3. Click **Persistent Stores**.

The Summary of Persistent Stores screen appears.

4. From the **New** drop-down list, select **Create JDBCStore**.

The Create a New JDBC Data Source screen appears.

5. Enter a name for the JDBC Data Source in the **Name** field.

For example, PersonJMSJDBCStore.

6. From the **Target** drop-down list, select **ohmpi_server-1 (migratable)**.

> **Note:** If you use a dynamic cluster, the Target is the entire cluster (ohmpi_cluster).

7. From the **Data Source** drop-down list, select **PersonJMSDataSource**.

8. Click **OK**.

9. Click **Activate Changes**.

Creating a JMS Module with JMS Connection Factory and JMS Topic

To create a JMS module with JMS connection factory and JMS topic:

1. Log in to the WebLogic administration console.

2. In the left pane, click on your domain (for example, domain1212) and expand **Services**.

3. Expand **Messaging** and click **JMS Modules**.

4. Click **Lock & Edit**.

5. Click **New**.

The Create JMS System Module screen appears.

6. Enter JMS server name in the **Name** field.

For example, PersonJMSModule.

7. Click **Next**.

8. In the **Targets** section, select the **ohmpi_cluster** check box.

9. Select the **All servers in the cluster** option.

10. Click **Next**.

11. Select the **Would you like to add resource to this JMS system module?** check box.

12. Click **Finish**.

13. In the Summary of Resources table, click **New**.

14. Select the **Connection Factory** option.

15. Click **Next**.

    The Connection Factory is created successfully.

16. Enter connection factory name in the **Name** field.

    For example, `PersonOutBoundSender`.

17. Enter JNDI Name in the **JNDI Name** field.

    For example, `jms/PersonOutBoundSender`.

18. Click **Next**.

    > **Note:** Do not change the default values.

19. Click **Finish**.

To create a JMS topic:

1. In the Create a New JMS System Module Resource screen, select the **Distributed Topic** option.

2. Click **Next**.

3. Enter JMS Topic name in the **Name** field.

    For example, `PersonTopic`.

4. Enter JNDI Name in the **JNDI Name** field.

    For example, `jms/PersonTopic`.

5. Click **Next**.

    > **Note:** Do not change the default values.

6. Click **Finish**.

    The JMS distributed topic is created successfully.

7. Click **Activate Changes**.

Starting Node Manager and Managed Servers

To start node manager and managed servers:

1. From the operating system prompt, start node manager by executing the following commands:

    - `<prompt>./startNodeManager.sh`

      or

    - `<prompt>startNodeManager.cmd`

      > **Note:** The command is located in the WebLogic domains/<your domain>/bin directory.

2. Start all managed servers from the WebLogic administration console.

3. Expand **Environment**.

4. Select **Cluster** and then select **ohmpi_cluster**.

5. Click **Control** and select all the managed server instances.

   For example, ohmpi_server-1 and ohmpi_server-2.

6. Click **Start**.

7. Click **Yes**.

8. Wait until all the managed servers are in the *Running* state.

   > **Note:** The page does not refresh automatically.

Deploying the MPI Application to the OHMPI Cluster

To deploy the MPI application to the OHMPI cluster:

1. Under Deployments, click **Install**.

2. Browse the MPI Application's deployable EAR file (for example, C:/NetBeansProjects/Person/dist/Person.ear).

3. Click **Next**.

4. Select the **Install this deployment as an application** option.

5. Click **Next**.

6. In the **Select deployment targets** section, select the **ohmpi_cluster** check box.

7. Select the **All servers in the cluster** option as the deployment target.

8. Click **Next**.

9. Select the **Copy this application onto every target for me** option.

10. Click **Finish**.

11. Click **Activate Changes**.

    > **Note:** The deployed application must be in the *Prepared* state.

12. Click **Lock & Edit**.

13. Select the application you deployed and click **Start**.

14. Select **Servicing All Requests**.

15. Click **Yes** to confirm.

    Wait until the state changes to *Active*.

Deploying a Proxy Web Application for Balancing HTTP Requests

To deploy a proxy web application for balancing HTTP requests:

1. Create a web application (for example, *LoadBalancerProxyWebApp*) in NetBeans for the WebLogic server with the following folder structure:

Figure 7–1   Folder Structure of LoadBalancerProxyWebApp
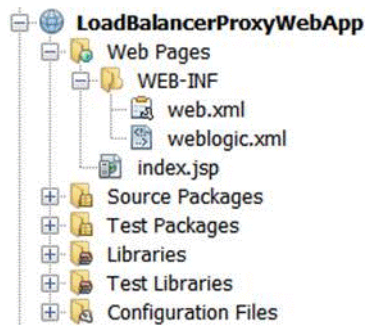
2. Modify the *weblogic.xml* file:

```
<?xml version="1.0" encoding="UTF-8"?>
<weblogic-web-app xmlns="http://www.bea.com/ns/weblogic/90">
<context-root>/</context-root>
</weblogic-web-app>
```

> **Note:** The `<context-root>` deployment parameter is set to `/`. This makes the proxy servlet as the default web application for the proxy server.

3. Modify the *web.xml* file.

> **Note:** Inside the *web.xml* file, the parameter value for WebLogicCluster must include all the servers on which the MIDM web application is deployed. The parameter value takes the format of *<host1>:<port1>*|*<host2>:<port2>*|…, where:
>
> - *host1* and *host2* are the host names of servers in the cluster.
> - *port1* and *port2* are the ports where the corresponding host is listening for HTTP requests.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
    <servlet>
        <servlet-name>HttpClusterServlet</servlet-name>

<servlet-class>weblogic.servlet.proxy.HttpClusterServlet</servlet-class>
        <init-param>
            <param-name>WebLogicCluster</param-name>
            <param-value>localhost:8001|localhost:8002</param-value>
        </init-param>
        <init-param>
            <param-name>verbose</param-name>
            <param-value>true</param-value>
        </init-param>
        <init-param>
            <param-name>DebugConfigInfo</param-name>
            <param-value>ON</param-value>
        </init-param>
```

```
            </servlet>
            <servlet-mapping>
                <servlet-name>HttpClusterServlet</servlet-name>
                <url-pattern>/</url-pattern>
                <url-pattern>*.jsp</url-pattern>
                <url-pattern>*.jspf</url-pattern>
                <url-pattern>*.jsf</url-pattern>
                <url-pattern>*.htm</url-pattern>
                <url-pattern>*.html</url-pattern>
                <url-pattern>*.do</url-pattern>
                <url-pattern>*.jpg</url-pattern>
                <url-pattern>*.gif</url-pattern>
                <url-pattern>*.css</url-pattern>
                <url-pattern>*.js</url-pattern>
                <url-pattern>*.png</url-pattern>
                <url-pattern>*.action</url-pattern>
            </servlet-mapping>
        </web-app>
```

4. Build the LoadBalancerProxyWebApp project.

5. Deploy LoadBalancerProxyWebApp on the system acting as the load balancer. (For example, *AdminServer*).

6. Access the MIDM application using the following URL:

   http://<LoadBalancer_IP_Address>:7001/PersonMIDM.

Modifying the EJB Client-Side JNDI Properties to Point to the Cluster Provider URL

To modify the EJB client-side JNDI properties to point to the cluster provider URL:

1. If there is an OHMPI client application, change its JNDI properties to use the new cluster provider URL. List all host and port numbers for all the managed servers in the cluster as a comma separated value.

   For example:

   ```
   java.naming.provider.url=t3://<host1>:<port1>,<host2>:<port2>,<host3>:<port3>
   ```

2. OHMPI provides Real-Time Loader, a Java EJB client, that takes advantage of the clustered EJBs for balancing load. To use a cluster provider URL, you can define the following property in the Real-time Loader's batch file (for example, *run-realtime-loader.bat*):

   ```
   set JAVA_NAMING_PROVIDER_
   URL=t3://<host1>:<port1>,<host2>:<port2>,<host3>:<port3>
   ```

8

Master Index Data Manager Login

This chapter provides information you need to set in order to log in to and run Master Index Data Manager (MIDM) on an application server.

This chapter includes the following section:

- Setting Up Servers to Log in to and Run the MIDM on page 8-1

Setting Up Servers to Log in to and Run the MIDM

The Master Index Data Manager runs on the WebLogic Server. The following section provides the information you need to set up the server for MIDM to run on it.

- Setting Up the User for MIDM Access Using WebLogic on page 8-1

Setting Up the User for MIDM Access Using WebLogic

In the following steps you create the MasterIndex.Admin and Administrator groups, and then create a new user within the two groups. Use the user you create for MIDM access using the WebLogic Admin Console.

1. On the left panel, under Domain Structure, expand **Services**, and then select **Security Realms**.

2. In the table on the Summary of Security Realms panel, click **myrealm**, which is the name of the realm.

   The Settings for myrealm panel appears.

3. Select the **Users and Groups** tab and then click **Groups**.

4. In the Groups table, click **New**.

5. In the **Name** field, type `MasterIndex.Admin` and click **OK**.

6. In the Groups table, click **New**.

7. In the **Name** field, type `Administrator` and click **OK**.

8. On the Settings for myrealm panel, select **Users and Groups** and then **Users**.

9. In the Users table, click **New**.

10. Type a name and a password for the new user you are creating and click **OK**.

11. Select **User Group**.

12. To add the two groups you created to the user you created, from the Available list, drag **MasterIndex.Admin** to the **Chosen** list, and then drag Administrator to the **Chosen** list.

9

Adding New Source Systems to an In-production MPI

To add new source systems to an in-production MPI, perform the following:

1. Add new source systems to the SBYN_SYSTEMS database table.

   For information on how to add new source systems to the SBYN_SYSTEMS table, see Chapter 3, "Oracle Healthcare Master Person Index Databases".

2. Add new source systems to the IHE_DOMAINS database table.

   > **Note:** This step is applicable only for IHE Application.

   For information on how to add new source systems to the IHE_DOMAINS table, see *Oracle Healthcare Master Person Index Working With IHE Profiles User's Guide*.

3. Update the MPI Project's `update.xml` file if the new source systems participate in weight-based survivor calculation while determining SBR. And, a special weight must be assigned to the new source system.

   For information on how to configure update.xml, see *Oracle Healthcare Master Person Index Configuration Guide*.

4. Update the MPI Project's `match-ext.xml` file if there is some special match configuration for the new source system being added.

   For information on how to configure `match-ext.xml`, see *Oracle Healthcare Master Person Index Configuration Guide*.

5. Perform one of the following:

   - If there are any changes to `update.xml` and `match-ext.xml`, regenerate the MPI Project. Then, rebuild and redeploy the MPI/IHE Project so that the project configuration changes take effect.

   - Else, restart the MPI/IHE Application.

6. Use OHMPI's Real-time Loader to load existing data records from the new source system to the MPI database.

   For information on how to use the Real-time Loader, see *Oracle Healthcare Master Person Index Real-time Loader User's Guide*.

> **Note:** Any configuration changes performed on an in-production MPI Application are not retroactive. That is, existing enterprise objects are not recalculated if they do not include a new system object from the new system being added based on the new configuration. All such enterprise objects are recalculated when updates are received from any source system, subsequent to the configuration changes.