Oracle® Healthcare Master Person Index

Working With IHE Profiles User's Guide

Release 2.0.5

E25244-04

May 2013

ORACLE®

Oracle Healthcare Master Person Index Working With IHE Profiles User's Guide, Release 2.0.5

E25244-04

Contents

4 Configuring an Application Server

5 Configuring and Using Audit Record Repository

6 Deploying the IHE Profiles Application

7 Monitoring and Maintaining Master Person Index Data

A HL7 Message Types Supported by IHE Within OHMPI

B Adding IHE Profile Support to User-defined MPI Object Model

Preface

This manual introduces users to Integrating the Healthcare Enterprise (IHE) and the Oracle Healthcare Master Person Index (OHMPI) IHE Profiles Application. IHE standards and profiles (PIX, PDQ, PAM, and ARR) help create, process, and manage electronic health records in secure patient cross-reference applications. They work in conjunction with native Health Level 7 (HL7) v2 and v3 messaging and transport standards which define how information is packaged and shared between systems. *Working With IHE Profiles* provides instructions on how to configure PIX, PDQ, PAM, and ARR components, create databases and runtime components, how to build and deploy an IHE Profiles Application project, and how to prepare the IHE Profiles Application to run data through it.

Audience

This document is intended for users of the OHMPI that want to create IHE Profiles Application projects.

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 Installation Guide*
- *Oracle Healthcare Master Person Index User's Guide*
- *Oracle Healthcare Master Person Index Data Manager User's Guide*
- *Oracle Healthcare Master Person Index Configuration Guide*
- *Oracle Healthcare Master Person Index Configuration Reference*

- *Oracle Healthcare Master Person Index Working With IHE Profiles User's Guide*
- *Oracle Healthcare Master Person Index WebLogic User's Guide*
- *Oracle Healthcare Master Person Index Standardization Engine Reference*
- *Oracle Healthcare Master Person Index Match Engine Reference*
- *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 Australia Patient Solution User's Guide*
- *Oracle Healthcare Master Person Index United Kingdom Patient Solution User's Guide*
- *Oracle Healthcare Master Person Index Message Processing Reference*
- *Oracle Healthcare Master Person Index Analyzing and Cleansing Data User's Guide*
- *Oracle Healthcare Master Person Index Loading the Initial Data Set User's Guide*
- *Oracle Healthcare Master Person Index Command Line Reports and Database Maintenance User's Guide*
- *Oracle Healthcare Master Person Index Release Notes*

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

Overview of the IHE Profiles Application

This chapter provides an overview of Oracle Healthcare Master Person Index (OHMPI) and the IHE Profiles Application. It also provides links to instructions in the following chapters that help you understand and use OHMPI and the IHE Profiles Application.

This chapter includes the following sections:

Introducing the IHE Profiles Application

OHMPI's IHE Profiles Application provides a flexible solution to healthcare integration needs. It uses processing logic based on the guidelines and standards put forth by the Integrating the Healthcare Enterprise (IHE) to assure compatibility with other vendors and healthcare organizations.

The IHE Profiles Application forms a specialized implementation of HL7 messaging that facilitates patient health information exchange, and supports both native HL7 v2 and HL7 v3 messaging. HL7 v2 is supported through the HL7 v2 server over the minimal lower layer protocol (MLLPV1 or MLLPV2). HL7 v3 is supported through SOAP 1.2 over HTTP.

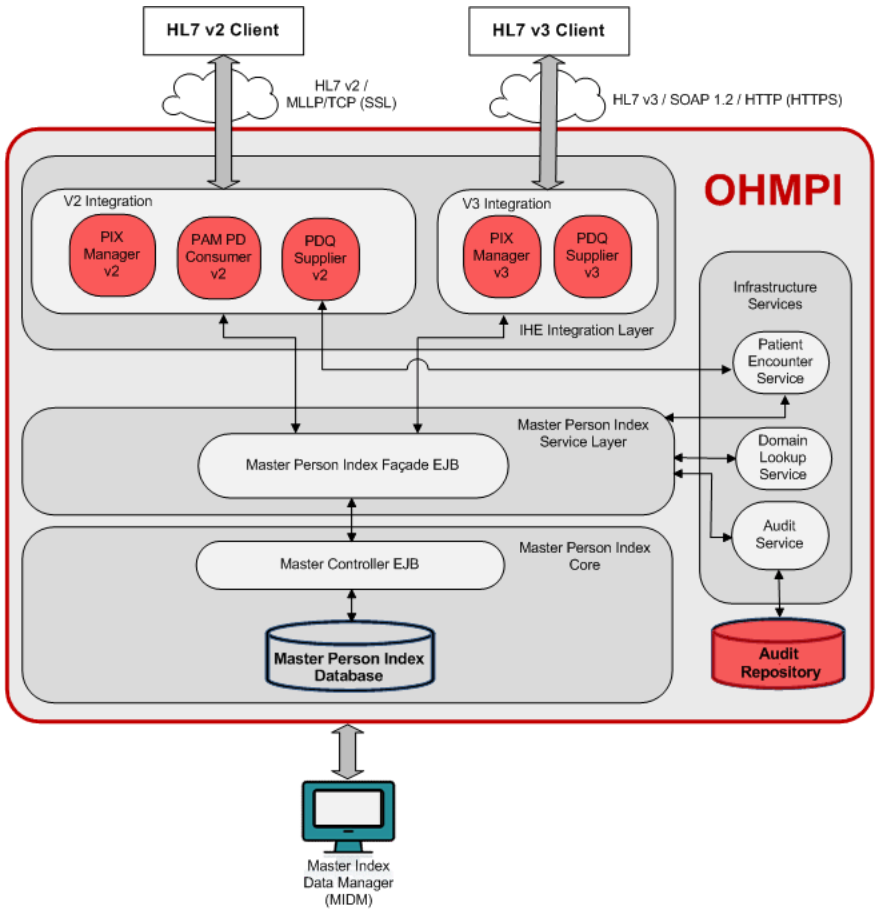
The IHE Profiles Application leverages the advanced standardization and matching algorithms of Master Person Index to cross-reference and uniquely identify the patients in your healthcare organization. Master Person Index provides a single complete view of the participants in your healthcare system and is able to quickly reconcile which information is associated with which patient.

The IHE Profiles Application processes messages based on the IHE IT infrastructure technical frameworks, which define how to process HL7 messages using existing standards when available. In compliance with these frameworks, the IHE Profiles Application generates and maintains an audit repository of all events processed by the manager. Information about the state of the IHE Profiles Application components is provided by a common logging, alerting, error handling, and reporting mechanism.

Understanding the Layers of the IHE Profiles Application

The IHE Profiles Application has a number of layers that function inside of OHMPI, which are illustrated in HL7 v2 and v3 Integration with OHMPI, and include:

Figure 1–1 HL7 v2 and v3 Integration with OHMPI



IHE Profiles Application Integration Layer

The IHE Integration layer adapts HL7 v2 and HL7 v3 messages to the simplified Java data model and service layer exceptions and vice versa. It also makes sure the message header properties (such as sending facility, receiving facility) are populated according to the specification.

The V3 Integration exposes OHMPI's functionality via SOAP version 1.2 over HTTP/HTTPS to IHE PIX/PDQ HL7 v3 clients. Its interface is defined by the IHE specification and the XSD/WSDL artifacts are provided by IHE.

The V2 Integration exposes OHMPI's functionality via TCP/IP-MLLP and HL7v2 based protocol to HL7 v2 clients.

Native HL7 Message Handling

The IHE Integration layer is HL7 v2 and v3 message-aware; that is, it can receive and validate incoming HL7 messages and handle standards-defined acknowledgements. HL7 v2 is supported through an HL7 v2 server that uses secured TCP/IP and Minimal Lower Layer protocol (MLLP). HL7 v3 is supported through SOAP messages transported over HTTP/HTTPS. Support for IHE Profiles in OHMPI is designed with the aim to implement the solution with minimal configuration level changes. This

layer works seamlessly in tandem with the Master Person Index layer to provide the appropriate patient identity resolution and demographic information responses.

Master Person Index Service Layer

The Master Person Index Service Layer consists of the core set of services mapping PIX/PDQ/PAM operations and data into OHMPI. The interface to this layer is IHE aligned. This means that the data structures are simplified versions of the HL7 message formats. The operations are analogous to the operations defined in IHE.

This layer also implements the audit requirements and domain resolution (namespace ID and universal ID mapping) service.

> **Note:** There is no protocol specific functionality in this layer. It implements the response behavior of PIX/PDQ/PAM from a functional point of view.

Master Person Index Core

This layer is the core patient identity resolution component that leverages OHMPI's powerful matching and standardization engines to uniquely identify and cross-reference patient objects based on a canonical patient object model. OHMPI's powerful, extensible, and configurable Match Engine compares complex data records containing a multitude of datatypes and calculates a global composite weight that reflects how closely the records match. From this, OHMPI builds a cross-index that is used to provide a single view of the patient information from multiple source systems in real time.

Infrastructure Services

As related to OHMPI and the IHE Profiles Application, the Audit Record Repository (ARR) is an audit server and repository which functions with a Domain Lookup Service that provides cross-referencing to identify a domain by domain ID (a.k.a. namespace ID) and/or universal ID, which in turn provides secure transmission, auditing, logging, alerting, error handling, and reporting for healthcare application systems. A supporting component is an audit service that generates audit messages and implements an IHE audit trail that is defined in the Audit Trail and Node Authentication (ATNA) Integration Profile.

Linking to IHE Profiles Application Tasks

This document includes several tasks that are optional and some that are required for a production environment. The tasks are listed by category below.

For Running Test Data

If you performed a complete installation of OHMPI, you are ready to work with the IHE Profiles Application and can start running test data and using the Master Index Data Manager (MIDM). See the following sections for instructions on running test data and monitoring the processed and data:

- Running Data Through the IHE Profiles Application on page 2

For a Production Environment

For a production environment, you need to configure certain variables unless you specified the correct values during installation, and each domain needs to be added to the IHE Profiles Application. Optionally, you can configure master person index processing.

Additional Steps to Set Up the IHE Profiles Application

There are additional steps you should consider performing before you begin using the IHE Profiles Application:

- Creating the IHE Profiles Application Databases and Tables on page 1
- Configuring Application Server Resources on page 4

2

Creating an IHE Profiles Application Project

This chapter provides procedures on how to create, configure, and build an IHE Profiles Application project. It also discusses the features of an IHE Profiles Application project Actions menu.

This chapter includes the following sections:

Creating an IHE Profiles Application Project

IHE design-time is a NetBeans IDE plug-in. The IHE wizard takes your input, such as database type, application server, and the V2 Service Port, and creates an IHE Profiles Application project and pre-configured Master Person Index (MPI) project. You are able to fine-tune the matching service of the master person index, and configure V2 Service Port of the IHE project. When the IHE project builds, the database scripts and three artifacts are generated. The IHE Create Database dialog allows you to create a database (if needed). Once the database is created you can deploy the IHE ear to the target server using the application server administration console. The steps to create the IHE Profiles Application project and deploy it are:

- Create an IHE Profiles Application project
- Configure the IHE Profiles Application project
- Build the IHE Profiles Application project
- Run the database script
- Configure the application server
- Deploy the IHE Profiles Application project on an application server

The IHE wizard requires the following input to create an IHE Profiles Application project and pre-configured MPI project. The wizard also sets the database type and application server information for both the IHE project and the pre-configured MPI project.

- Project name
- Project location

- Application server
- Database
- V2 Service Port

To Create an IHE Profiles Application Project

1. From the NetBeans IDE, click **File** and then choose **New Project**.

   The New Project dialog appears.

2. In the Projects section of the Choose Project panel, choose **IHE Profiles Application** and click **Next**.

   The New IHE Profiles Application dialog appears.

3. In the Name and Location panel, do the following:
   - In the Project Name field, type a name for your project (for example, **IHEProject1**).
   - In the Project Location field, browse to the path where you want to create the IHE project.

     The Project Folder field populates with the path and name of your project.
   - From the Server list, choose a server (**GlassFish 2.1** or **WebLogic 11gR1**).

     Note: While the OHMPI Installer installs GlassFish 2.1, WebLogic 11gR1is a separate installation. See the *Oracle Healthcare Master Person Index WebLogic User's Guide*.

   - In the Database list, choose a database (for example, **Oracle**).
   - In the V2 Service Port field, type the port number (for example, **4447**).
   - Click **Finish**.

Configuring the Master Person Index Project

You can configure several features of the Master Person Index to customize how data is matched and processed through the IHE Profiles Application. The type and quality of data stored for a patient varies from system to system and from organization to organization. The Master Person Index takes this into account and allows you to tailor the processing for your specific data requirements. For example, you may know that some systems contain more accurate data than others. You can weight those systems higher when determining which values will populate the single best record (SBR). Also, some fields might contain more accurate data than others and would thus be given a higher match weight than other, less accurate fields.

You can configure the following components of the Master Person Index:

- Patient demographic queries, matching queries, and queries performed from the MIDM.
- Standardization and matching rules, including which fields to standardize, which fields to match, a range of match weights to assign to each field, and the type of algorithm to use to match each field.
- Filters for match results (for example, you can configure the Master Person Index to ignore default values).

- The single best record and how it is formed.
- The appearance of the Master Index Data Manager (MIDM).
- Field validations.

In addition, you can customize logic for the underlying process of the master person index. For complete information on configuring the Master Person Index, see *Oracle Healthcare Master Person Index Configuration Guide*.

If you modify the configuration of the Master Person Index, you need to do the following:

- Rebuild and redeploy the IHE Profiles Application Project. See Creating an IHE Profiles Application Project.

Note: You cannot modify an object model in an IHE-MPI application.

Using the IHE Project Actions Menu

The Actions menu that you access by right-clicking an IHE Profiles Application project in the left panel of NetBeans IDE provides you with the necessary tools to manage a project.

- **Build** - Builds an IHE Profiles Application project and generates its artifacts.
- **Clean and Build** - Removes generated artifacts from an IHE Profiles Application project and then rebuilds the project and generates new artifacts.
- **Clean** - Removes generated artifacts from an IHE Profiles Application project.
- **Create Database** - Opens the Create Database dialog (see Creating the Database and Tables Automatically).
- **Set as Main Project** - Sets a selected IHE Profiles Application project as the main project.
- **Open Required Project** - Opens a pre-configured Master Person Index project.
- **Close** - Closes an IHE Profiles Application project.
- **Properties** - Opens the Project Properties - <*Project_Name*> dialog, where you can change the HL7 V2 Service Port and Web Service Security Setting.

Configuring an IHE Profiles Application Project

The information provided in this section is the same for GlassFish and WebLogic. Before building an IHE Profiles Application project you need to configure it. This includes:

- Editing the following configuration files to change IHE service properties.
  - `hl7message.xml`
  - `hl7service.xml`
  - `services.properties`
- Using the Master Person Index editor to fine-tune matching configuration. See the *Oracle Healthcare Master Person Index Configuration Guide* for information about using the editor.

- If the web service endpoints of the IHE Profiles Application needs to be secured with TLS, check the **Secure Application** check box. This will disable the plain HTTP access to the web service endpoints, and only allow HTTP over TLS access with mutual authentication. See To Configure Secure PIXv2 Manager and PDQv2 Supplier and Configuring Secure Web Services on GlassFish Enterprise Server v2.1.1 for additional information on secure application configuration.

To Configure hl7message.xml

`hl7message.xml` defines HL7 v2 native message properties according to the HL7 specification.

- **validateMSH**

  An indicator of whether MSH segment validation is enabled.

- **seqNumEnabled**

  An indicator of whether sequence number protocol is enabled.

- **processingId**

  The processing Id value of the MSH-11 segment in the received message is validated when validateMSH is enabled. MSH-11 is used to indicate whether a message is processed as defined in the HL7 processing rules. The processing Id options include the following:

  – **D** - Debugging

  – **T** - Training

  – **P** - Production

- **sendingApplicationNamespaceId**

  The sending application (MSH-03 segment) is used to create an acknowledgement or response message. This is used to help identify the application from other participating applications within the network enterprise.

- **sendingApplicationUniversalId**

- **sendingApplicationUniversalIdType**

- **sendingFacilityNamespaceId**

  The sending facility (MSH-04 segment) is responsible for creating an acknowledgement or response message. This is used to help further identify participating facilities within the network enterprise.

- **sendingFacilityUniversalId**

- **sendingFacilityUniversalIdType**

- **enncodingCharacters**

  The four encoding characters used to create an acknowledgement or response message. This attribute contains the four characters in the following order:

  – **Component separator**

  – **Repetition separator**

  – **Escape character**

  – **Subcomponent separator**

  The recommended value is ^~\& (that is, ASCII 94, 126, 92, and 38, respectively).

- **fieldSeparator**

  The separator is used to separate between the segment Id and the first real field. This value defines the character that is used as a separator for the rest of the message. Enter the value as a decimal ASCII number ranging from 1 to 127. The default value is 124 which is the character "|".

- **softwareVendorOrganization**

  The Software Vendor Organization field (SFT-1-Software Vendor Organization) identifies the vendor who is responsible for maintaining the application. This property only applies to HL7 versions 2.5 relevant IHE integration profile messages.

- **softwareCertifiedVersionOrReleaseNumber**

  The Software Certified Version or Release Number is the value of the HL7 segment SFT-02. The current software version number or release number for the sending system helps to provide a more complete profile of the application that is sending or receiving the HL7 messages. This property only applies to HL7 versions 2.5 relevant IHE integration profile messages.

- **softwareProductName**

  The name of the software product that submitted the transaction is the value of the HL7 segment SFT-03. The software product name is a key component for identifying the sending application. This property only applies to HL7 versions 2.5 relevant IHE integration profile messages.

- **softwareBinaryId**

  The Software Binary Id is the value of HL7 segment SFT-04. This property is available starting with HL7 version 2.5. Software Binary Ids are issued by a vendor for each unique software version instance. These Ids are used to differentiate between differing versions of the same software. Identical Primary Ids indicate that the software is identical at the binary level, but configuration settings may differ. This property only applies to HL7 versions 2.5 relevant IHE integration profile messages.

- **softwareProductInformation**

  The software product identification information is the value of the HL7 segment SFT-05. This may include a description of the software application, configuration settings, modifications made to the software, and so forth. This information is used for diagnostic purposes to help identify the application software. This property only applies to HL7 versions 2.5 relevant IHE integration profile messages.

- **softwareInstallDate**

  The Software InstallDate is the value of HL7 segment SFT-06. This is the date on which the submitting software was installed at the sending site. The install date can provide key information in regard to the behavior of an application. The date format is YYYYMMDDHHSS. This property only applies to HL7 versions 2.5 relevant IHE integration profile messages.

To Configure hl7service.xml

`hl7service.xml` defines hl7 v2 server runtime communication protocol properties.

The runtime communication protocol properties of the OHMPI HL7 v2 server specifies the protocol information for the external HL7 system to connect to the OHMPI HL7 v2 server and vice versa.

- **url**

  A URL used for an external system to connect to the OHMPI HL7 v2 server. The format of the URL is: hl7://<*host*>:<*port*>.

- **threadCount**

  The maximum number of processing threads.

- **connectionTimeout**

  The idle time in seconds for a connection to enforce a to be closed action.

- **maxConnectRetries**

  The maximum number of connection retires the outbound connector will attempt before executing the recourse action.

  – **Value**: The value is presented in the format: n1;n2, n1;n2, ..., where n1 is the number of retry attempts to connect, and n2 is the interval in seconds between retry attempts. If the value of n1 is -1, it indicates that the number of retry attempts is indefinite.

  – **Action**: Suspend and Error.

- **TimeToWaitForAResponse**

  The amount of time the outbound connector waits for a response from the external system before executing the configured recourse action.

  – **Value**: An integer indicating the time interval in seconds.

  – **Action**: Resend, Reset, and Suspend.

- **maxNakSent**

  The maximum number of NAKs the server sends to the client before executing the recourse action. A negative acknowledgment is sent from the HL7 inbound connection when the received message from client is invalid. An invalid message, according to the HL7 specification, is one that fails MSH segment validation or has an invalid sequence number.

  – **Enabled**: true or false

  – **Value**: An integer indicating the appropriate maximum number.

  – **Action**: Suspend and Reset.

- **maxCannedNakSent**

  The maximum number of Canned NAKs the server sends to the client before executing the configured recourse action. This communication control deals with the case where the message received from the client is invalid as per the HL7 specification rules for that particular message. A Canned HL7 NAK is created when a read error occurs, or when a message cannot be identified as an HL7 message.

  – **Enabled**: true or false

  – **Value**: An integer indicating the appropriate maximum number.

  – **Action**: Suspend and Reset.

- **mllpv1**

The MLLP Version 1 properties configure the Minimal Lower Layer Protocol (MLLP) version 1 property.

- **llpType**: MLLPv1 indicates Minimal Lower Layer Protocol v1.0.
- **StartBlockCharacter**: The start block character value in a decimal ASCII number from 1 to 127 specifies a block start. Unless there is a conflict, the value should be ASCII VT which is decimal 11.
- **EndBlockCharacter**: The end block character value in decimal ASCII number from 1 to 127 specifies a block end. To be strictly compliant with the HL7 standard, this parameter must be set to a carriage return which is decimal 13.
- **EndDataCharacter**: The end data character value in decimal ASCII number from 1 to 127 specifies a data end. To be strictly compliant with the HL7 standard, the value should be a carriage return indicated by decimal 28.

- **mllpv2**

  The MLLP Version 2 properties configure the Minimal Lower Layer Protocol (MLLP) version 2 property.

  - **llpType**: MLLPv2 indicates Minimal Lower Layer Protocol v2.0.
  - **StartBlockCharacter**: The start block character value in a decimal ASCII number from 1 to 127 specifies a block start. Unless there is a conflict, the value should be ASCII VT which is decimal 11.
  - **EndBlockCharacter**: The end block character value in decimal ASCII number from 1 to 127 specifies a block end. To be strictly compliant with the HL7 standard, this parameter must be set to a carriage return which is decimal 13.
  - **EndDataCharacter**: The end data character value in decimal ASCII number from 1 to 127 specifies a data end. To be strictly compliant with the HL7 standard, the value should be a carriage return indicated by decimal 28.
  - **RetryCountOnNak**: The maximum configured number of attempts to send a message after receiving a negative acknowledgement.
  - **RetryInterval**: The duration in milliseconds to wait between each retry attempt.
  - **timeToWaitForAckNak**: The duration in milliseconds to wait for a commit positive or negative acknowledgement.
  - **hllp**
    - **llpType**: HLLP indicates Hybrid Lower Layer Protocol.
    - **StartBlockCharacter**: The start block character value in a decimal ASCII number from 1 to 127 specifies a block start. Unless there is a conflict, the value should be ASCII VT which is decimal 11.
    - **EndBlockCharacter**: The end block character value in decimal ASCII number from 1 to 127 specifies a block end. To be strictly compliant with the HL7 standard, this parameter must be set to a carriage return which is decimal 13.
    - **EndDataCharacter**: The end data character value in decimal ASCII number from 1 to 127 specifies a data end. To be strictly compliant with the HL7 standard, the value should be a carriage return indicated by decimal 28.
    - **checksumEnabled**: An indicator of whether the HLLP Checksum feature is enabled. This can only be enabled if the LLP Type is set to HLLP.

The actions supported by the IHE OHMPI HL7 v2 server can include the following values:

- **Reset**

  The server closes the connection with the HL7 external system and throws an alert.

- **Resend**

  The server component resends the last sent message to the HL7 external system.

- **Suspend**

  The server closes the connection with the HL7 external system, suspends the connection from processing the messages, and throws an alert.

To Configure Secure PIXv2 Manager and PDQv2 Supplier

To secure IHE Profiles Application you need to enable SSL support, including Mutual authentication.

> **Note:** This configuration step is optional.

To configure secure PIXv2 Manager and PDQv2 Supplier you need to configure `hl7service.xml`:

- **url**

  Define a URL used for secured connection. The formation of the URL is: `hl7s://<host>:<port>`.

- **keyStore**

  Define a keyStore file full path where private keys in JKS format are stored.

- **keyStorePassword**

  Define a password to access to the keyStore.

- **trustStore**

  Define a trustStore file full path where trusted certificates in JKS format are stored.

- **trustStorePassword**

  Define a password to access to the trustStore.

- **keyManagerKeyStorePassword**

  Define a password for KeyManager.

- **clientAuthentication**

  Set **true** for requiring client authentication, or mutual authentication. Set **false** for only requiring server authentication.

- **protocols**

  Define supported SSL protocols. You can define a list of different protocols delimited by "," (that is, you can define "SSLv3,TLSv1").

- **cipherSuites**

  Define supported SSL cryptographic algorithms. You can define a list of different cipher suites delimited by "," (that is, you can define "TLS_RSA_WITH_AES_128_CBC_SHA,SSL_RSA_WITH_NULL_SHA").

To import keys and certificates into KeyStore or TrustStore, you need to do:

- If your certificate or key is X509 format, transform X509 certificates and keys to pkcs#12 format using openssl tool:

```
openssl pkcs12 -export -out my_keystore.pkcs12 -in my_server_cert.pem
-inkey my_server_key.pem
```

- Import the pkcs#12 certificate into a JKS keystore using keytool:

```
keytool -importkeystore -deststorepass changeit -destkeystore my_
keystore.jks -srckeystore my_keystore.pkcs12 -srcstoretype PKCS12
-srcstorepass changeit
```

- Import the public keys into the trust store using keytool:

```
keytool -import -noprompt -trustcacerts -file my_client_cert.pem
-keystore my_truststore.jks -alias client
```

To Configure services.properties

What follows is a partial list of user-configurable properties for IHE profiles and HL7 v3 web services. As this release does not have a nice UI support for editing these properties, you need to manually edit the services.properties file before building the IHE project.

Open the services.properties file from within NetBeans IDE and make the appropriate edits to the following.

- PIXv3-Manager-Receiver-Device-Namespace-ID=PIX_X_REF_MGR_ORACLE
- PIXv3-Manager-Receiver-Device-Universal-ID=1.3.6.1.4.1.21367.13.30.20
- PIXv3-Manager-Receiver-Device-Organization-Namespace-ID=ORACLE
- PIXv3-Manager-Receiver-Device-Organization-Universal-ID=1.3.6.1.4.1.21367.13.50.21
- PIXv3-Manager-Sender-Device-Namespace-ID=PIX_X_REF_MGR_ORACLE
- PIXv3-Manager-Sender-Device-Universal-ID=1.3.6.1.4.1.21367.13.10.18
- PIXv3-Manager-Sender-Device-Organization-Namespace-ID=ORACLE
- PIXv3-Manager-Sender-Device-Organization-Universal-ID=1.3.6.1.4.1.21367.13.50.21
- PDQ-Continuation-Timeout=60
- PDQ-Continuation-Batch-Size=20
- Enable-PIX-Update-Notification=true
- Enable-EUID-Query=false
- OHMPI-Assigning-Authority-Namespace-ID=OHMPI Assigning Auth
- OHMPI-Assigning-Authority-Universal-ID=1.3.6.1.4.1.21367.2010.1.2.300

Identifies the PIXv3 Manager Receiver Device (This also serves as the identification for PDQv3 Supplier Receiver Device.):

- PIXv3-Manager-Receiver-Device-Namespace-ID
- PIXv3-Manager-Receiver-Device-Universal-ID

Identifies the PIXv3 Manager Receiver Organization (This also serves as the identification for PDQv3 Supplier Receiver Device Organization.):

- PIXv3-Manager-Receiver-Device-Organization-Namespace-ID

- PIXv3-Manager-Receiver-Device-Organization-Universal-ID

Identifies the PIXv3 Manager Sender Device (for PIX Update Notification):

- PIXv3-Manager-Sender-Device-Namespace-ID
- PIXv3-Manager-Sender-Device-Universal-ID

Identifies the PIXv3 Manager Sender Organization (for PIX Update Notification):

- PIXv3-Manager-Sender-Device-Organization-Namespace-ID
- PIXv3-Manager-Sender-Device-Organization-Universal-ID

Sets the default Patient Demographics Query Continuation timeout (in seconds):

- PDQ-Continuation-Timeout

Sets the default Patient Demographics Query Continuation batch size:

- PDQ-Continuation-Batch-Size

Sets whether to enable PIX v2/v3 Update Notification:

- Enable-PIX-Update-Notification

Sets whether to enable PIX/PDQ Query by EUID and PIX/PDQ Query for EUID:

- Enable-EUID-Query

Identifies the OHMPI Assigning Authority, that is, the EUID Domain:

- OHMPI-Assigning-Authority-Namespace-ID
- OHMPI-Assigning-Authority-Universal-ID

Note: Both OHMPI-Assigning-Authority-Namespace-ID and OHMPI-Assigning-Authority-Universal-ID need to be defined properly only when the EUID Query is enabled. The OHMPI Assigning Authority Domain, that is, the EUID Domain, also needs to be added into the IHE_DOMAINS table in this case.

Building an IHE Profiles Application Project

This section provides the steps to build a project, along with the differences between GlassFish and WebLogic.

To Build an IHE Profiles Application Project

1. In the left panel of NetBeans IDE, right-click the project icon (for example, **IHEProject1**).

   An Actions menu appears.

2. To build an IHE Profiles Application project, click **Build** on Actions menu.

When you build the IHE Profiles Application project for GlassFish, two artifacts are created:

- `<Project_Name>.ear`: The IHE Profiles Application.
- `hl7v2.zip`: The HL7 v2 lifecycle module.

The hl7v2.zip is an HL7 v2 server package that includes all artifacts required to run the IHE OHMPI HL7 v2 server and HL7 v2 lifecycle model which manages the server. You unzip it to the location where you run the server. The unzipped folder includes:

hl7v2

- config

  The config sub folder includes all the configuration files.

- lib

  The lib sub folder includes all the required jar files.

3

Creating MPI and IHE Databases and Tables

This chapter provides procedures that lead you through creating Master Person Index and IHE Profiles Application databases and tables.

This chapter includes the following section:

Creating the IHE Profiles Application Databases and Tables

After installing OHMPI you need to create the Master Person Index, IHE Profiles Application database tables. You must create these tables before you can work with the IHE Profiles Application. You can either create the databases and tables through OHMPI design-time (which is recommended), or create them manually.

> **Note:** Creating a database is optional; if desired you can create your own database.

To create databases and table manually, use:

To create databases and tables using OHMPI design-time, use:

To configure optional connecting information, use:

Creating Master Person Index Databases and Tables for MySQL Manually

Before you begin, make sure you have MySQL installed as a service with remote host access enabled. Perform the following steps to create all databases:

To Create the Master Person Index Database (MySQL)

The IHE Profiles Application solution currently supports MySQL 5.1.x. Before you begin this process, be sure you have MySQL installed as a service with remote host access enabled.

- Create a password for the root user.
- Run the following command against the root user:

```
grant all privileges on *.* to 'root'@'%' with grant option;
```

1. Create a schema in your MySQL environment named **mpi** by running the following command:

```
create database mpi;
```

2. Create a user for the Master Person Index tables by running the following command:

```
grant alter,alter routine,create,create
routine,delete,drop,execute,index,insert,select,trigger,update,create
temporary tables on mpi.* to '<USER_NAME>' identified by
'<PASSWORD>'
```

3. Open `systems.sql` in a text editor, and add the HL7 systems with which you will be sharing data.

> **Note:** See Step 6 for the location of this file.

For more information about modifying this file, see *Oracle Healthcare Master Person Index User's Guide*.

4. Open `codelist.sql` in a text editor, and add any common code information you need for the master person index system.

> **Note:** See Step 6 for the location of this file.

For more information about code lists and modifying this file, see *Oracle Healthcare Master Person Index User's Guide*.

> **Note:** This step is optional. The default configuration does not use common code tables. Adding this feature will cause additional validations to be performed against incoming data.

5. Log in to the **mpi** schema as the user you created above.
6. Run the following SQL files against the **mpi** schema (*<project_name>*`\ihe-mpi\src\DatabaseScript`). The `create.sql` file must be run first.
   - `create.sql`
   - `systems.sql`
   - `codelist.sql`

Continue to To Create the IHE Profiles Application Database Tables (MySQL).

To Create the IHE Profiles Application Database Tables (MySQL)

Before you begin, complete To Create the Master Person Index Database (MySQL).

1. Navigate to the directory to which you created the IHE Profiles Application projects and then navigate to the *<project_name>*`\src\DatabaseScripts` folder.

- `create_ihe_ohmpi_tables.sql` creates all the IHE Profiles Application tables.
- `create_ihe_ohmpi_sample_data.sql` creates sample domain configuration data.
- `clean_ihe_ohmpi_tables.sql` cleanses all the records from IHE Profiles Application tables and MPI tables.

2. Open `create_ihe_ohmpi_sample_data.sql` in a text editor, comment out all existing insert statements, and add any systems you added to the `systems.sql` file in To Create the Master Person Index Database (MySQL).

   You need to insert the corresponding namespace ID, universal ID, universal ID type and a description for each new system; for example:

   ```
   insert into IHE_DOMAINS (NAMESPACEID, UNIVERSALID, UNIVERSALIDTYPE,
   DESCRIPTON) values ('HOSPITAL1', '1.4.5.2.6.2.23455', 'ISO', 'HOSPITAL1
   DESCRIPTION');
   ```

3. (Optional) To configure patient update notification, see Maintaining Subscriptions to Patient Updates.
4. Save and close the file.
5. Log in to the **mpi** schema as the user you created in To Create the Master Person Index Database (MySQL).
6. Run the following scripts against the **mpi** schema.
   - `create_ihe_ohmpi_tables.sql`
   - `create_ihe_ohmpi_sample_data.sql`

Creating Master Person Index Databases and Tables for Oracle Manually

Before you begin, make sure you have Oracle installed as a service with remote host access enabled. Perform the following steps to create all databases:

- To create the IHE Profiles Application Database Tables (Oracle)

To Create the Master Person Index Database (Oracle)

1. Create an Oracle environment and user for the Master Person Index tables by running the following command:

   ```
   grant CREATE SESSION, CREATE TABLE, CREATE TRIGGER, CREATE PROCEDURE,
   UNLIMITED TABLESPACE, CREATE TYPE to <USER_NAME> identified by <PASSWORD>
   ```

2. Open `systems.sql` in a text editor, and add the HL7 systems with which you will be sharing data.

   Note: See Step 5 for the location of this file.

   For more information about modifying this file, see *Oracle Healthcare Master Person Index User's Guide*.

3. Open `codelist.sql` in a text editor, and add any common code information you need for the master person index system.

   Note: See Step 5 for the location of this file.

For more information about code lists and modifying this file, see *Oracle Healthcare Master Person Index User's Guide*.

> **Note:** This step is optional. The default configuration does not use common code tables. Adding this feature will cause additional validations to be performed against incoming data.

4. Log in to the **mpi** schema as the user you created above.
5. Run the following SQL files against the **mpi** schema (`<project_name>\ihe-mpi\src\DatabaseScript`). The `create.sql` file must be run first.
   - `create.sql`
   - `systems.sql`
   - `codelist.sql`

Continue to To Create the IHE Profiles Application Database Tables (Oracle).

To Create the IHE Profiles Application Database Tables (Oracle)

Before you begin, complete To Create the Master Person Index Database (Oracle).

1. Navigate to the directory to which you created the IHE Profiles <*project_name*> Application projects, and then navigate to the `\src\DatabaseScripts` folder.
   - `create_ihe_ohmpi_tables.sql` creates all the IHE Profiles Application tables.
   - `create_ihe_ohmpi_sample_data.sql` creates sample domain configuration data.
   - `clean_ihe_ohmpi_tables.sql` cleans all the records from IHE Profiles Application tables and MPI tables.
2. Open `create_ihe_ohmpi_sample_data.sql` in a text editor, comment out all existing insert statements, and add any systems you added to the `systems.sql` file in To Create the Master Person Index Database (Oracle).

   You need to insert the corresponding namespace ID, universal ID, universal ID type and a description for each new system; for example:

   ```
   insert into IHE_DOMAINS (NAMESPACEID, UNIVERSALID, UNIVERSALIDTYPE,
   DESCRIPTON) values ('HOSPITAL1', '1.4.5.2.6.2.23455', 'ISO', 'HOSPITAL1
   DESCRIPTION');
   ```

3. (Optional) To configure patient update notification, see Maintaining Subscriptions to Patient Updates.
4. Save and close the file.
5. Log in to the **mpi** schema as the user you created in To Create the Master Person Index Database (Oracle).
6. Run the following scripts against the **mpi** schema.
   - `create_ihe_ohmpi_tables.sql`
   - `create_ihe_ohmpi_sample_data.sql`

Creating the Database and Tables Automatically

1. Right click the name of your IHE Profiles Application project in the left panel of the NetBeans IDE (for example, **IHEProject1**) and choose **Create Database**.

   The Create Database dialog appears. Here you provide the credentials for an existing Oracle or MySQL server.

2. With the Root Connection tab selected, do the following to configure the database:
   - **Server Host**: The name of the server where the database is located.
   - **Root User** - The login name to the database.
   - **Root Password**: The password used to log in to the database.
   - **Server Port**: The port number of the server where the database is located.
   - **SQL URL**: The URL for the database.
3. After the Connection requirements for the database are set, click **Test Connection**.
4. When satisfied with the Connection settings, click the **User-Schema** tab.
5. With the User-Schema tab selected, do the following to configure the database User:
   - View the User for the database from the list.
   - Accept the default password or type a new password.
   - Click **Apply**.
6. Click the **Database Scripts** tab to view the scripts and then do the following:
   - To run the scripts, click **Run**.
   - After the scripts complete running, click **Close**.

Maintaining Subscriptions to Patient Updates

The IHE Profiles Application was designed to be implemented with minimal required customizations. You can customize and extend the functionality of OHMPI's connectivity information for a product environment.

In order for a PIX Consumer to receive PIX Update Notifications from a source domain, add a subscription record to the IHE_PIXUPDATE_SUBSCRIPTIONS table. For example,

```
insert into IHE_PIXUPDATE_SUBSCRIPTIONS (CONSUMERID, SOURCENAMESPACEID,
ACTIVE) values ('CONSUMER1', 'IHEGHCADT', 'T');
```

where `CONSUMER1` identifies a PIX Consumer that receives updates and `IHEGHCADT` is the source domain that sends updates.

For each PIX Consumer that is interested in receiving PIX Update Notifications, a record describing the PIX Consumer's endpoint is required to be added to the IHE_PIXCONSUMER_ENDPOINTS table first.

Make sure the following is correct when configuring PIX Consumer Endpoints:

- The identifier of the PIX Consumer. This needs to match the `CONSUMERID` in the `IHE_PIXUPDATE_SUBSCRIPTIONS` table.
- The endpoint type (either "HL7V2" or "HL7V3").
- The LLPType (applicable only if the endpoint type is "HL7V2").

- The receiving URL for the PIX Consumer.
- The name of the HL7 application or device.
- The name of the facility or organization (organization is optional).

The following is an example describing a PIXv2 Consumer:

```
insert into IHE_PIXCONSUMER_ENDPOINTS (CONSUMERID, ENDPOINTTYPE, LLPTYPE,
ADDRESS, APPL_DEV, FACL_ORG, DESCRIPTION) values ('CONSUMER1', 'HL7V2',
'MLLPv1', 'hl7://localhost:9906', 'SUNPIXPDQRAD', NULL, 'Test domain');
```

The following is an example describing a PIXv3 Consumer:

```
insert into IHE_PIXCONSUMER_ENDPOINTS (CONSUMERID, ENDPOINTTYPE, LLPTYPE,
ADDRESS, APPL_DEV, FACL_ORG, DESCRIPTION) values ('CONSUMER2', 'HL7V3',
NULL, 'http://localhost:8080/PIXConsumer2_Service/PIXConsumer2',
'1.2.840.114350.1.13.99997.2.3412', NULL, 'Test domain');
```

4

Configuring an Application Server

You need to configure the lifecyle modules for the application server you use. You also need to configure IHE-required application server resources, such as JDBC resources and JMS resources.

This chapter includes the following sections:

Configuring GlassFish Enterprise Server v2.1.1 and Lifecycle Modules

For GlassFish Enterprise Server configuration information, go to `http://docs.oracle.com/cd/E19575-01/821-0185/` to see the *Sun GlassFish Enterprise Server v2.1.1 Administration Guide*.

Lifecycle Modules

Lifecycle modules, which initiate at server start up, allow you to run Java-based tasks within a GlassFish Enterprise Server environment. When you use GlassFish, you need to install and configure one lifecycle module:

- HL7V2 Server Lifecycle Module

> **Note:** You must install and configure both modules on the GlassFish Server before you can deploy the IHE EAR file.

You can install the lifecycle modules from the command line or the admin console.

To Install the HL7 v2 Lifecycle Module From the Command Line

To install IHE HL7 v2 lifecycle module to your target application server, perform the following:

1. Unzip `hl7v2.zip` to a directory of your choice (for example, your OHMPI installation directory. The unzipped `hl7v2` directory is your HL7v2 server home).
2. Start your GlassFish server (if it is not already running).
3. Navigate to your GlassFish installation's `bin` folder.
4. From the command prompt, run **asadmin** as

```
asadmin create-lifecycle-module

--classname oracle.hsgbu.ohmpi.ihe.hl7v2.management.HL7ServiceGFManager

--classpath <your HL7 v2 server folder root
path>\lib\ihe-mpi-hl7v2-management.jar

--loadorder 300

--property oracle.hsgbu.ohmpi.ihe.hl7v2.home=[your HL7 v2 server folder
root path]

hl7v2
```

Note: If you used a colon in the property, you need to escape it with an "\\", for example:

<*property_ABC*>=C\:\\hl7v2Client

rather than

<*property_ABC*>=C:\hl7v2Client

5. Set environment variable OHMPI_IHE_HL7V2_HOME to point to your HL7v2 server root:set OHMPI_IHE_HL7V2_HOME=<*your HL7v2 Server root path*>.
6. Restart your GlassFish server.

To Configure the HL7 v2 Lifecycle Module From the GlassFish Admin Console

1. Log in to the GlassFish Admin Console.
2. Click the **Lifecycle Modules** node in the left panel, then click the **New** button in the right panel under Lifecyle Modules.

   The New Lifecyle Module dialog appears. We are going to assume that we are creating the Audit Repository LifeCycle Module to complete this procedure.
3. In the Name field, type the name of for your module (for example, hl7v2).

   It is a good idea to choose a name that represents the module. "Arr" stands for Audit Record Repository.
4. In the Class Name field, type oracle.hsgbu.ohmpi.ihe.hl7v2.management.HL7ServiceGFManager.

   Note: The class name can only contain alphanumeric, underscore, dash, and dot characters.
5. In the Classpath field, type the path to the *<your HL7v2 Server root path>* /lib/ihe-mpi-hl7v2-management.jar file.

   Note: You can leave this field blank if the class is already in the server's class path.
6. (Optionally) In the Load Order field, set the order in which you want the lifecycle modules to load when the application server starts up.

Note: Modules with smaller integers load earlier.

7. (Optionally) In the Description field, describe the lifecycle module.
8. Select the **Status** check box.
9. To prevent an instance startup if a module load fails, select the **On Load Failure** check box.
10. Click **OK** to save your selections.

    A newly created lifecycle module with the name you gave it appears under Lifecycle Modules in the left panel (for example, `hl7v2`).
11. Click the newly created lifecycle module to edit it. Open it in the Edit Lifecycle Module dialog. Edit it and use the "Add Property" button to add one property:

    It appears in the Edit Lifecycle Module dialog, with the fields populated.
12. Under Additional Properties, click the **Add Property** button, and do the following to configure your hl7v2 home directory:
    - For Name type `oracle.hsgbu.ohmpi.ihe.hl7v2.home`.
    - For Value type *<your hl7v2 server folder root path>*.
13. Set environment variable `OHMPI_IHE_HL7V2_HOME` to point to your HL7v2 server `root:set OHMPI_IHE_HL7V2_HOME=`*<your HL7v2 Server root path>*.
14. Restart GlassFish to load the lifecycle module.

Administering Lifecycles From the Command Line

After you have installed and configured lifecycles on GlassFish you can administer them from the command line.

To List Lifecycles From the Command Line

1. Start your application server (if it is not already running).
2. Run the subcommand **list-lifecycle-modules**.

Note: Run the command in remote mode.

Example 4–1 Running the list-lifecycle-modules Subcommand

```
asadmin> list-lifecycle-modules
hl7v2
Command list-lifecycle-modules executed successfully
```

To Delete a Lifecycle Module From the Command Line

1. Start your application server (if it is not already running).
2. Run the subcommand **list-lifecycle-modules** to obtain a list of your current lifecycles.

Note: Run the command in remote mode.

3. Run the subcommand **delete-lifecycle-module** to delete a lifecycle module.

Example 4–2 Running the delete-lifecycle-module Subcommand

```
asadmin> delete-lifecycle-module obsolete_hl7v2
Command delete-lifecycle-module executed successfully
```

Configuring Application Server Resources

Before running OHMPI with IHE Profiles you must configure application server resources.

Creating the JDBC Connection Pools (MySQL)

The IHE Profiles Application maintains connections to several databases to comply with IHE requirements for monitoring, audit logging, and master data management.

To Create the JDBC Connection Pools (MySQL)

The following procedure outlines the steps to create a connection pool. The configuration information for each connection pool you need to create is listed after the procedure.

1. Log in to the GlassFish Admin Console.

   You can access the console from the Services window in NetBeans IDE.

2. In the left portion of the Admin Console, expand **Resources**, expand **JDBC**, and then select **Connection Pools**.
3. On the Create Connection Pool page, click **New**.
4. In the Name field, enter a name for the connection pool. See the informal table at the end of this section for information about the connection pools.

   Note: *To complete steps 4 and 5 see the informal table at the end of this section for the four connection pools to be configured.*

5. In the Resource Type field, select the Java class for the type of transactions being processed.
6. In the Database Vendor field, select **MySQL**.
7. Click **Next**.
8. In the DataSource Classname field, accept the default class.
9. Modify the **Pool Settings**, **Connection Validation**, and **Transaction** properties according to your business practices.

10. In the additional properties section, enter the values for the database. Be sure to enter the following properties at a minimum (you might need to create some of these properties).
    - **URL**: The URL that points to the database. The syntax of the URL is `jdbc:mysql://<server>:<port>/<DatabaseName>`.
      - *server*: The name of the server where the database is located.
      - *port*: The port number of the database. For MySQL, the default port is **3306**.
      - *DatabaseName*: The name of the database schema.
    - **User**: The login ID for database user.
    - **Password**: The password for the above user.
    - **DatabaseName**: The name of the database schema.

    Note: These properties are mandatory. Also note that the database name in the URL must match the name given to the DatabaseName property.

    Create the connection pools listed below. The database vendor for all pools is MySQL. The port number and server name depend on your MySQL environment, and the URL you enter depends on these values. The user and password properties are the ones you defined for each database schema in Creating the IHE Profiles Application Databases and Tables.

    Note: You do not have to use the connection pool names given here. These are the default names given by the installer.

Connection Pool Name	Resource Type	Database Name	Purpose
cpPatientXA	The type corresponding to the type of transaction processed by the master person index. The default is javax.sql.XADataSource.	mpi	Connecting to the Master Person Index database.
cpPatientSequenceXA	The type corresponding to the type of transactions that are distributed, either within the application or across applications The default is javax.sql.XADataSource.	mpi	Connecting to the Master Person Index sequence table.
cpPIXDomainLU	javax.sql.DataSource	mpi	Connecting to the Domain Lookup table.
cpPIMPendingLinksXA	javax.sql.XADataSource	mpi	Connecting to the PAM/PIM Pending Links table.

Creating the JDBC Connections Pools (Oracle)

The IHE Profiles Application maintains connections to several databases to comply with IHE requirements for monitoring, audit logging, and master data management.

To Create the JDBC Connection Pools (Oracle)

The following procedure outlines the steps to create a connection pool. The configuration information for each connection pool you need to create is listed after the procedure.

1. Log in to the GlassFish Admin Console.

   You can access the console from the Services window in NetBeans IDE.

2. In the left portion of the Admin Console, expand **Resources**, expand **JDBC**, and then select **Connection Pools**.

3. On the Create Connection Pool page, click **New**.

4. In the Name field, enter a name for the connection pool.

   Note: *To complete steps 4 and 5 see the informal table at the end of this section for the four connection pools to be configured.*

5. In the Resource Type field, select the Java class for the type of transactions being processed.

6. In the Database Vendor field, select **Oracle**.

7. Click **Next**.

8. In the DataSource Classname field, accept the default class.

9. Modify the Pool Settings, Connection Validation, and Transaction properties according to your business practices.

10. In the additional properties section, enter the values for the database. Be sure to enter the following properties at a minimum (you might need to create some of these properties).

    - **URL**: `jdbc:oracle:thin:@<server>:<port>/<SID>`
      - *server*: The name of the server where the database is located.
      - *port*: The port number of the database. For Oracle, the default port is **1521**.
      - *SID*: The name of the database schema.
    - **User**: The login ID for database user.
    - **Password**: The password for the above user.
    - **DatabaseName**: The name of the database schema.

    Note: These properties are mandatory. Also note that the database name in the URL must match the name given to the DatabaseName property.

Create the connection pools listed below. The database vendor for all pools is Oracle. The port number and server name depend on your Oracle environment, and the URL you enter depends on these values. The user and password

properties are the ones you defined for each database schema in Creating the IHE Profiles Application Databases and Tables.

Note: You do not have to use the connection pool names given here. These are the default names given by the installer.

Connection Pool Name	Resource Type	Database Name	Purpose
cpPatientXA	The type corresponding to the type of transaction processed by the master person index. The default is javax.sql.XADataSource.	mpi	Connecting to the Master Person Index database.
cpPatientSequenceXA	The same as above.	mpi	Connecting to the Master Person Index sequence table.
cpPIXDomainLU	javax.sql.DataSource	mpi	Connecting to the Domain Lookup table.
cpPIMPendingLinksXA	javax.sql.XADataSource	mpi	Connecting to the PAM/PIM Pending Links table.

Creating the JDBC Resources for MySQL and Oracle

The following procedure outlines the steps to create a JDBC resource. The configuration information for each resource you need to create is located after the procedure.

To Create the JDBC Resources for MySQL and Oracle

1. In the left portion of the Administration Console, expand **Resources**, expand **JDBC**, and then select **JDBC Resources**.
2. On the Create JDBC Resource page, click **New**.
3. In the JNDI Name field, enter a unique name for the JDBC resource.
4. In the Pool Name field, select the name of a JDBC connection pool.
5. In the Status field select the **Enabled** check box.
6. Click **OK**.

   Create the JDBC Resources listed below, which correspond to the connection pools you created earlier. Note that several of the JNDI names are case-sensitive.

JNDI Name	Pool Name
jdbc/PatientDataSource	cpPatientXA
jdbc/PatientSequenceDataSource	cpPatientSequenceXA
jdbc/PIXDomainLUDataSource	cpPIXDomainLU
jdbc/PIMPendingLinksDataSource	cpPIMPendingLinksXA

Creating JMS Destination Resources and Connection Factory

The IHE Profiles Application uses JMS topics to make outbound patient notification updates available to the domains that subscribe to those updates. The Oracle Healthcare Master Person Index generates outbound messages for patient updates in XML format to the PatientTopic destination. The PIX Update Manager that handles update notifications make the information available to either HL7 v2 or HL7 v3 destinations according to the IHE framework.

You create the JMS artifacts manually as described below.

To Create the JMS Destination Resources and Connection Factory

1. In the left portion of the Admin Console, expand Resources, expand JMS Resources, and then select Connection Factories.
2. On the New JMS Connection Factory page, click **New**.
3. Enter the following information:
   - **JNDI Name** - jms/PatientOutBoundSender (this name is case-sensitive)
   - **Resource Type** - javax.jms.TopicConnectionFactory
4. Create the following additional properties and enter the property values:
   - **UserName** - A JMS user name. The default user name is **admin**.
   - **Password** - A password for the above user name. The default password is **admin**.
5. Click **OK**.
6. In the left portion of the Administration Console, select **Destination Resources** under JMS Resources.
7. On the New JMS Destination Resource page, click **New**.
8. Use this page to create the following topics:

JNDI Name	Resource Type	Physical Destination Name	Purpose
jms/PatientTopic	javax.jms.Topic	PatientTopic	Updates to the Master Person Index patient records are written to this topic.
jms/PixUpdateNotificationTopic	javax.jms.Topic	PixUpdateNotificationTopic	HL7 v2 and v3 updates to patient records are processed through a Java EE Message Driven Bean and written to this topic.

Creating User Accounts for MIDM Access

You create user accounts for MIDM access using the GlassFish Admin Console. There are two predefined user groups for the MIDM, and you can define additional user groups each with different permissions. For more information about configuring user groups (roles) for MIDM, see *Oracle Healthcare Master Person Index Data Manager User's Guide*.

To Create User Accounts for MIDM Access

1. Log on to the GlassFish Administration Console.

2. In the left navigation panel, expand **Configuration**, expand **Security**, and then expand **Realms**.
3. Select **File**.
4. On the Edit Realm page, select **Manage Users**.

   The File Users page appears.
5. On the File Users page, select **New**.
6. In the User ID field, enter a name for the user.
7. In the Group List field, for MIDM users, type `MasterIndex.Admin` and any of the predefined user groups or user groups that you defined (for example, `Administrator`). Separate multiple roles with a comma (no space).
8. Enter a password for the user in the New Password field.
9. In the Confirm New Password field, enter the password again.
10. Click **OK**.

Configuring Secure Web Services on GlassFish Enterprise Server v2.1.1

Note: This configuration step is optional.

In order to support secure IHE Profiles applications on the HL7 v3 (web service) side, the application server needs to have correct keystore and truststore setup.

Suppose *XXX*`.key.pem` and *XXX*`.cert.pem` are your private key and the corresponding public certificate in X509 format, respectively. Also suppose `ca.cert.pem` is the public certificate of the CA that signed your certificate, and `cacerts.jks` is the existing GlassFish truststore under *<Glassfish>*`/domains/`*<domain_name>*`/config`. To import keys and certificates into GlassFish's keystore and truststore, use the following steps:

1. Create a temporary working directory that contains the above four files.
2. Transform X509 certificates and keys to pkcs#12 format using openssl tool:

   ```
   openssl pkcs12 -export -inkey XXX.key.pem -in XXX.cert.pem -out
   XXX.pkcs12 -name s1as
   ```

   - Enter Export Password: `changeit`.
   - Confirm the Export Password.
3. Import the pkcs#12 certificate into a brand new JKS keystore using keytool:

   ```
   keytool -importkeystore -srckeystore XXX.pkcs12 -srcstoretype pkcs12
   -srcstorepass changeit -srcalias s1as -destkeystore keystore.jks
   -deststoretype jks -deststorepass changeit
   ```
4. Import the public certificate of the CA into the existing truststore using keytool:

   ```
   keytool -importcert -file ca.cert.pem -keystore cacerts.jks -storepass
   changeit
   ```
5. (Optional) Back up your existing GlassFish application server's keystore (`keystore.jks`) and truststore (`cacerts.jks`) under *<Glassfish>*`/domains/`*<domain_name>*`/config`.

6. Copy the new keystore (`keystore.jks`) and truststore (`cacerts.jks`) generated in steps 3 and 4 above to `<Glassfish>/domains/<domain_name>/config`.
7. Start GlassFish if it is not started.
8. Log in to the Admin Console.
9. Select **Application Server** on the left side tree, then select **JVM Settings**, and then **JVM Options** on the right hand side, make sure the keyStore and trustStore properties are set correctly, for example:
   - `-Djavax.net.ssl.keyStore=${com.sun.aas.instanceRoot}/config/keystore.jks`
   - `-Djavax.net.ssl.trustStore=${com.sun.aas.instanceRoot}/config/cacerts.jks`
10. Select **Configuration** on the left hand side tree, then **HTTP Service**, **HTTP Listeners**, **http-listener-2**. On the right hand side, in the **Edit HTTP Listener** tab, make sure that **Listener** and **Security** are both enabled and the **Listener Port** is properly configured (for example, **8181**).
11. In the **SSL** tab, make sure **TLS** is enabled and the **Certificate NickName** matches the alias you used in step 2 (for example, **s1as**).
12. If the **Client Authentication** check box is not checked, then only server authentication is required. To enable client authentication (or mutual authentication) for all applications deployed on the application server, check the **Client Authentication** checkbooks. Note that even if the client authentication is not enabled here, individual application deployed on the application server can still require client authentication on a per-endpoint basis. Refer to Creating an IHE Profiles Application Project on how to secure web service endpoints in an IHE application.

    Note: Even if the client authentication is not enabled here, an individual application deployed on the application server can still require client authentication on a per-endpoint basis. See Creating an IHE Profiles Application Project for information on how to secure web service endpoints in an IHE Profiles Application.

13. If specific ciphersuites are required during SSL/TLS handshake, select the desired ciphersuites from the **Available Common Ciphersuites** list and click **Add** to place them in the **Selected Common Ciphersuites** list.
14. If any changes need to be made during steps 9 through 13, save the changes and restart the application server.

5

Configuring and Using Audit Record Repository

The Audit Record Repository (ARR), which includes an audit server and an audit repository, is part of the Internet Protocol Suite that deals with the transmission of data. Specifically related to OHMPI and the IHE Profiles Application, ARR provides secure transmission and auditing for healthcare application systems. The major components of the Audit Record Repository include:

- Audit Trail and Node Authentication (ATNA) Integration Profile
  - http://wiki.ihe.net/index.php?title=ATNA

  which is built on top of the following:
- Security Audit and Access Accountability Message XML Data Definitions for Healthcare Applications
  - http://tools.ietf.org/html/rfc3881
- The Syslog Protocol
  - http://tools.ietf.org/html/rfc5424
- Transmission of Syslog Messages over Transport Layer Security (TLS)
  - http://tools.ietf.org/search/rfc5425
- Transmission of Syslog Messages over User Datagram Protocol (UDP)
  - https://tools.ietf.org/html/rfc5426

> **Note:** The above links open documents that deal with the Internet Protocol Suite, specifically "Internet Official Protocol Standards" (STD1) as related to ARR. They provide critical technical information about secure transmission of data over the internet, including node authentication and an audit trail. It is recommended that you read them.

This chapter includes the following sections:

Installing the Audit Record Repository

When you install the runtime (GlassFish) using the OHMPI Installer, ARR is installed in a directory named **arr** at the same directory level as GlassFish. This installation is automatic and requires no user intervention.

Running the Audit Record Repository Server

You run ARR from the command line using the Audit Record Repository command line script:

- For Windows: `arr.bat`
- For any of the UNIX platforms: `arr.sh`

For help using the command line script, type `> arr -help`.

Description of the Audit Record Repository Script

`usage: arr -propertyfile` <*propertyfile*> `-command` <*command*> <*...args*>

Use the above script to start and test an instance of ARR (use `CTRL^C` to stop the server).

Commands

- `create-tables`

  Creates the required ARR database tables and sequences.

  – **Options**

    * `-arr.persistence_unit_name`

      The name of the javax persistence unit defined in `persistence.xml`.

    * `-arr.jdbc_driver`

      The JDBC database driver type, for example:

      - **MySQL**: `com.mysql.jdbc.Driver`

      - **Oracle**: `oracle.jdbc.OracleDriver`

    * `-arr.jdbc_url`

      The JDBC database url.

    * `-arr.jdbc_username`

      The JDBC database username.

    * `-arr.jdbc_password`

      The JDBC database password.

- `drop-and-create-tables`

  Drops and recreates the ARR database tables and sequences.

  – **Options**

    * `-arr.persistence_unit_name`

      The name of the javax persistence unit defined in `persistence.xml`.

    * `-arr.jdbc_driver`

The JDBC database driver type, for example:

- **MySQL**: `com.mysql.jdbc.Driver`

- **Oracle**: `oracle.jdbc.OracleDriver`

* `-arr.jdbc_url`

  The JDBC database url.

* `-arr.jdbc_username`

  The JDBC database username.

* `-arr.jdbc_password`

  The JDBC database password.

- `parse-audit-msg`

  Tests the validity of an audit message.

  – **Options**

  * `-arr.input_file`

    A file containing an audit message.

- `parse-syslog-msg`

  Tests the validity of a syslog message.

  – **Options**

  * `-arr.input_file`

    A file containing a syslog message.

- `send-tls-msg`

  Sends a syslog message to an ARR supporting TLS.

  – **Options**

  * `-arr.input_file`

    A file containing a syslog message.

  * `-arr.hostname`

    The hostname of the syslog server.

  * `-arr.port`

    The port of the syslog server.

  * `-arr.keystore`

    The client keystore.

  * `-arr.keystore_password`

    The client keystore password.

  * `-arr.truststore`

    The client truststore.

  * `-arr.truststore_password`

    The client truststore password.

  * `-arr.keymanager_keystore_password`

The client keymanager keystore password.

- `send-udp-msg`

  Sends a syslog message to ARR supporting UDP.

  - **Options**

    * `-arr.input_file`

      A file containing a syslog message.

    * `-arr.hostname`

      The hostname of the syslog server.

    * `-arr.port`

      The port of the syslog server.

- `start-tls-server`

  Starts a TLS ARR running on a given port.

  - **Options**

    * `-arr.port`

      The port to listen on (6514 is the standard port for syslog over TLS).

    * `-arr.persistence_unit_name`

      The name of the javax persistence unit defined in `persistence.xml`.

    * `-arr.jdbc_driver`

      The JDBC database driver type, for example:

      - **MySQL**: `com.mysql.jdbc.Driver`

      - **Oracle**: `oracle.jdbc.OracleDriver`

    * `-arr.jdbc_url`

      The JDBC database url.

    * `-arr.jdbc_username`

      The JDBC database username.

    * `-arr.jdbc_password`

      The JDBC database password.

    * `-arr.keystore`

      The server keystore.

    * `-arr.keystore_password`

      The server keystore password.

    * `-arr.truststore`

      The server truststore.

    * `-arr.truststore_password`

      The server truststore password.

    * `-arr.keymanager_keystore_password`

      The server keymanager keystore password.

- `start-udp-server`

  Starts an UDP ARR running on a given port.

  – **Options**

    * `-arr.port`

      The port to listen on (514 is the standard port for syslog over UDP).

    * `-arr.persistence_unit_name`

      The name of the javax persistence unit defined in `persistence.xml`.

    * `-arr.jdbc_driver`

      The JDBC database driver type, for example:

      - **MySQL**: `com.mysql.jdbc.Driver`

      - **Oracle**: `oracle.jdbc.OracleDriver`

    * `-arr.jdbc_url`

      The JDBC database url.

    * `-arr.jdbc_username`

      The JDBC database username.

    * `-arr.jdbc_password`

      The JDBC database password.

Examples of Commands

- `create-tables`

  ```
  > arr -propertyfile arr.properties -command create-tables
  ```

- `drop-and-create-tables`

  ```
  > arr -propertyfile arr.properties -command drop-and-create-tables
  ```

- `parse-audit-msg`

  ```
  > arr -propertyfile arr.properties -command parse-audit-msg -arr.input_
  file test_audit_msg.txt
  ```

- `parse-syslog-msg`

  ```
  > arr -propertyfile arr.properties -command parse-syslog-msg
  -arr.input_file test_syslog_msg.txt
  ```

- `send-tls-msg`

  ```
  > arr -propertyfile arr.properties -command send-tls-msg -arr.hostname
  localhost -arr.input_file test_syslog_msg.txt
  ```

- `send-udp-msg`

  ```
  > arr -propertyfile arr.properties -command send-udp-msg -arr.hostname
  localhost -arr.input_file test_syslog_msg.txt
  ```

- `start-tls-server`

  ```
  > arr -propertyfile arr.properties -command start-tls-server
  ```

- `start-udp-server`

  ```
  > arr -propertyfile arr.properties -command start-udp-server
  ```

Examples of Property Files

MySQL Property File Example

```
arr.persistence_unit_name=jpa
arr.jdbc_driver=com.mysql.jdbc.Driver
arr.jdbc_url=jdbc:mysql://localhost:3306/arr
arr.jdbc_username=arruser
arr.jdbc_password=arrpass

arr.port=514

arr.keystore=keystore/arr_keystore.jks
arr.keystore_password=changeit
arr.truststore=keystore/arr_truststore.jks
arr.truststore_password=changeit
arr.keymanager_keystore_password=changeit
```

Oracle Property File Example

```
arr.persistence_unit_name=jpa
arr.jdbc_driver=oracle.jdbc.OracleDriver
arr.jdbc_url=jdbc:oracle:thin:@localhost:1521:XE
arr.jdbc_username=arruser
arr.jdbc_password=arrpass

arr.port=514

arr.keystore=keystore/arr_keystore.jks
arr.keystore_password=changeit
arr.truststore=keystore/arr_truststore.jks
arr.truststore_password=changeit
arr.keymanager_keystore_password=changeit
```

Configuring the Audit Client

In order for the IHE Profiles Application to send audit records to the Audit Record Repository Server, you have to properly configure an OHMPI audit client. The OHMPI audit client's configuration file is named by `ohmpi-audit-client.properties`. Depending on the type of the Application Server, this property file's location is different. For GlassFish, this property file is under *<GlassFish>*`\lib`; for WebLogic, this file is under `domains\<`*domain_name*`>\lib`.

There are two properties defined in `ohmpi-audit-client.properties`: `auditHost` and `auditPort`. They are the host name and port number of the ARR server, respectively. The default value for `auditHost` is **localhost**, and it needs to match wherever the ARR server is running. The default value for `auditPort` is **514**, and it needs to match the value of `arr.port` defined in `arr\bin\arr.properties`.

6

Deploying the IHE Profiles Application

This chapter provides procedures on how to deploy the IHE Profiles Application to the GlassFish application server and how to run HL7 v2 and HL7 v3 sample data through the IHE Profiles Application.

This chapter includes the following section:

Deploying an IHE Profiles Application

Deploying an IHE Profiles Application project to a GlassFish application server through the server's web Admin Console. This starts the IHE service.

To Deploy an EAR File from an Admin Console

1. From left panel of the NetBeans IDE, right-click an **IHE Profiles Application project**. from the Actions Menu,

   The IHE Project Actions menu appears.

2. Click **Clean and Build**.
3. Start GlassFish.

   The Admin Console appears.

4. In the left panel, choose **Applications**, select **Enterprise Applications**, and then click **Deploy** under Deployed Enterprise Applications in the right panel.
5. Browse to the EAR file of your project, select it, and click **OK**.

To Deploy an EAR File from the Command Line

1. From the Project Explorer of the NetBeans IDE, right-click an **IHE Profiles Application project**,

   The IHE Project Actions menu appears.

2. Click **Clean and Build**.
3. Open the command prompt.
4. Navigate to the <*app_server*>`/bin` directory.
5. Run **asadmin deploy** <*directory_path_t*o>`/IHEMPI.ear`.

Running Data Through the IHE Profiles Application

Oracle provides both HL7 v2 and HL7 v3 sample data for you to run through the PIX/PDQ system to help you get started with the applications. To enter the HL7 v2 sample messages, you need an HL7 simulator. There are several available on the internet. The HL7 v3 data is provided in the form of test cases that are run from the soapUI project.

The sample data illustrates several scenarios for data processing. You add new records, update existing records, cause assumed matches in the Master Person Index, perform PIX queries for associated local identifiers, perform demographic queries, and perform merges. The data was designed to be run in a specific order to get the expected results from PIX and PDQ queries.

Before you can run these tests, you need to run a database script to add system information to the midm database, as described below. Without this data, the sample tests will fail.

Note: You can modify the HL7 v2 sample files and the HL7 v3 test input files to insert additional records into the master person index database. If you modify the HL7 v2 sample files, be sure to use an editor that does not automatically convert `\r` to `\r\n`, such as UltraEdit.

Use the steps in the following procedures to add the required systems to the database, and then to run HL7 V2 and V3 data.

To Add Required Systems to the Database

1. Navigate to the location where you installed OHMPI, and then to `samples\databasescripts\<database_type>`.
2. Log into the **mpi** database from a SQL prompt or editor as the **mpi** user.
3. Run the `clean_ihe_ohmpi_sample_data.sql` file against the database.
4. Run the `create_ihe_ohmpi_sample_data.sql` file against the database.

To Run HL7 V2 Sample Data

Before you begin, make sure that your application server is running, your OHMPI IHE enterprise application is configured and deployed correctly, your HL7 V2 service lifecycle model is deployed and started correctly, your HL7 v2 client simulator is configured and starts correctly. You can use any HL7 v2 compliant client to send and receive HL7 v2.5 and v2.3.1 sample data. Configuring the HL7 v2 client simulator is specific to the type of the simulator you choose. Generally, you should only need to configure its environment to use HL7 v2 listening URL, which is defined in the `oracle.hsgbu.ohmpi.ihe.hl7v2.url` property for the OHMPI IHE HL7 V2 lifecycle module (by default `hl7://localhost:4447`), or in the `hl7service.xml` configuration file.

Note: Each group of sample data files has a readme file that you should review to learn how each sample data file should be processed and what expected results should be.

---

1. Navigate to the OHMPI installation directory and then to `samples\hl7v2`.

This folder contains a complete set of HL7 v2 sample data files. It also contains subdirectories that each contains a use case. A use case contains a subset of data files that perform specific functions. It contains database scripts for preparing the database to run HL7 v2 sample data. A readme file in the use case directory explains what the use case is and how to run sample data files and the expected results.

2. To process additional records into the system, first backup your sample data files and modify the message data as needed, then use the **HL7 v2 simulator** to complete the procedure.
3. After processing sample data files from a use case, you can:
   - View the response message or the acknowledge message.
   - Review records from an OHMPI master person index using the Master Index Data Manager (MIDM).
   - Review audit records and message tracing, content, and extensions.

To Run HL7 V3 Sample Data

Under the `samples\hl7v3` directory of your OHMPI installation, there is a soapUI sample project that you can use to test your PIXv3 and PDQv3 web services. A readme file explains the test case scenarios.

Before you begin, make sure the application server is running, your OHMPI IHE Profile Application has been deployed, and soapUI has been installed properly.

> **Note:** To install soapUI, go to `http://soapui.org`, click the **Download Free Open Source** button, download the latest version (currently 3.6.1) for your platform (for example, soapui-3.6.1-windows-bin.zip). Unzip the zip file after downloading it.

For best results, run the tests in the order they are displayed in the project. This will show you a full spectrum of PIX v3/PDQ v3 processing. If the tests are run out of order additional errors will occur because the response from the Master Person Index will be different.

1. Adjust web service URLs in the sample soapUI project if necessary.

   The soapUI sample project provided is based on an IHE Profile Application deployed on GlassFish with default configuration, that is, the PIX Manager v3 and PDQ Supplier v3 web services are assumed to be running at `http://localhost:8080/PIXManager_Service/PIXManager` and `http://localhost:8080/PDQSupplier_Service/PDQSupplier`, respectively. If your IHE Profile Application is running on a different host or a different port on GlassFish, or if your application is running on WebLogic, then you need to adjust the above two URLs in the soapUI sample project accordingly. To adjust the URLs, open the `samples\hl7v3\pixpdqv3-samples-soapui-project.xml` in your favorite editor, replace all instances of the above two URLs based on the following instructions:

   - For Glassfish, the web service URLs should be in the following format:
     - `http://<host_name>:<port_number>/PIXManager_Service/PIXManager`
     - `http://<host_name>:<port_number>/PDQSupplier_Service/PDQSupplier`

- For WebLogic, the web service URLs should be in the following format: (the default port number on WebLogic is **7001**):
  - `http://<host_name>:<port_number>/PIXManager_Soap12/PIXManager_Service`
  - `http://<host_name>:<port_number>/PDQSupplier_Soap12/PDQSupplier_Service`

2. To start soapUI, go to the `${SOAPUI_HOME}\bin` directory and invoke `soapui.bat`.
3. To import the PIXV3/PDQV3 soapUI project, click **File**, select **Import Project**, and then select `pixpdqv3-samples-soapui-project.xml` from the `samples\hl7v3` directory.
4. Expand the **PIX and PDQ v3 Samples** node in the project tree.
5. Double click the **TestSuite-PIXv3** or **TestSuite-PDQv3** node to show the Test Suite Editor.
6. Click the **Run** button to run all the test cases in the test suite.
7. After running a test suite, you can:
   - Check whether the tests passed or failed within soapUI.
   - Review records using the Master Index Data Manager (MIDM).
   - Review audit records and message tracing.

7

Monitoring and Maintaining Master Person Index Data

Use the Master Index Data Manager to monitor and maintain your master person index data.

This chapter includes the following section:

Using the Master Index Data Manager to Monitor and Maintain Master Person Index Data

The Master Index Data Manager provides the following functions so you can maintain the patient records in the master person index database:

- View patient records
- Compare patient records
- Merge and unmerge patient records
- Compare records that are flagged as possible duplicates
- Resolve or merge potential duplicate records
- Review and reverse automatic record matches made by the master person index
- Create reports on master person index data and statistics, including records added or updated, potential duplicates, merged and unmerged records, and so on

Logging in to the Master Index Data Manager

The MIDM is accessed through a web browser, and requires you to log in before you can access any patient information. The URL for the MIDM is:

- `http://`*`host`*`:`*`http_port`*`/`*`app_name`*`MIDM`

where

- *`host`* is the name of the server machine.
- *`http_port`* is the HTTP port number of the application server.
- *`app_name`* is the name of the master person index application.

The default URL for MIDM is:

- `http://localhost:8080/PatientMIDM` (GlassFish)

- http://localhost:7001/PatientMIDM (WebLogic)

To Log in to the MIDM

Before you begin:

- Make sure you have a login ID and password.
- Make sure the application server is running.

1. Launch a web browser.
2. In the Address field, enter the appropriate URL.

   The login page appears.
3. In the upper right portion of the page, select the language for the MIDM display.
4. Enter your user ID and password in the appropriate fields.
5. Click Login.

   The initial page appears. (By default, the initial page is the Record Details page, but this is configurable.)

A

HL7 Message Types Supported by IHE Within OHMPI

This appendix provides a summary of all HL7 message type transactions for PIX v2.3.1, PDQ v2.5, PAM v2.5, PEM v2.5, PDVQ v2.5, PIX v3, and PDQ v3 that Oracle Healthcare Master Person Index supports.

Supported HL7 v2.3.1, v2.5, and v3 Message Types in OHMPI

The tables in this appendix summarize all the message types of PIX v2.3.1, PDQ v2.5, PAM v2.5, PEM v2.5, PDVQ v2.5, PIX v3, and PDQ v3 transactions that are supported by IHE within OHMPI. See the following tables for information about the IHE Profile name, HL7 message type, and the action performed by the message transactions:

Table A–1 PIX v2.3.1 Supported Message Types

IHE Profile Name	HL7 Message Type	Action
ITI-8: Patient Identity Feed (v2.3.1)	ADT A01 ADT_A01	Add Patient
	ADT A04 ADT_A01	
	ADT A05 ADT_A05	
	ADT A08 ADT_A01	Update Patient
	ADT A40 ADT_A39	Merge Patients
	ACK	Acknowledgement
ITI-9: PIX Query (v2.5)	QBP Q23 QBP_Q21	Query Message
	RSP K23 RSP_K23	Query Response
ITI-10: PIX Update Notification (v2.5)	ADT A31 ADT_A05	Update Notification
	ACK	Acknowledgement

Table A–2 PDQ v2.5 Supported Message Types

IHE Profile Name	HL7 Message Type	Action
ITI-21: Patient Demographics Query (v2.5)	QBP Q22 (QBP_Q21)	Find Candidates Query
	RSP K22 (RSP_K22)	Query Response
	QBP Q22 (QBP_Q21)	Query Continue
	QCN J01 QCN_J01[1]	Query Cancel
	ACK	Cancel Acknowledgement

[1] The specification requires three components, while the sample data only contains the first two.

Table A–3 PAM v2.5 Supported Message Types

IHE Profile Name	HL7 Message Type	Action
ITI-30: Patient Identity Management (v2.5)	ADT A28 ADT_A05	Create Patient
	ADT A31 ADT_A05	Update Patient
	ADT A40 ADT_A39	Merge Patients
	ADT A47 ADT_A30	Change Patient Identifier List
	ADT A24 ADT_A24	Link Patient
	ADT A37 ADT_A37	Unlink Patient

Table A–4 PEM v2.5 Supported Message Types

IHE Profile Name	HL7 Message Type	Action
ITI-31: Patient Encounter Management (v2.5)	ADT A01 ADT_A01	Admit inpatient
	ADT A04 ADT_A01	Register Outpatient
	ADT A03 ADT_A03	Discharge patient
	ADT A08 ADT_A01	Update patient information
	ADT A40 ADT_A39	Merge patient identifier list
	ADT A11 ADT_A09	Cancel admit/visit
	ADT A13 ADT_A01	Cancel Discharge
	ADT A05 ADT_A05	Pre-admit patient
	ADT A38 ADT_A38	Cancel pre-admit
	ADT A06 ADT_A06	Change patient class to inpatient
	ADT A07 ADT_A06	Change patient class to outpatient
	ADT A02 ADT_A02	Transfer patient
	ADT A12 ADT_A12	Cancel transfer

Unusual Situation with Outpatient and Inpatient Encounters

The recording of an outpatient encounter when an inpatient encounter is already open (and not ended) for the same patient is not usual. The OHMPI implementation is made on the following lines:

- Discard the outpatient encounter if inpatient encounter is already open.
- Record the outpatient encounter if the inpatient encounter has ended. (that is, the patient has been discharged).

Table A–5 PDVQ v2.5 Supported Message Types

IHE Profile Name	HL7 Message Type	Action
ITI-22: Patient Demographics And Visit Query (v2.5)	QBP_ ZV1 (QBP_ZV1)	Find Candidates From Visit Query
	RSP ZV2 (RSP_ZV2)	Visit Query Response
	QCN J01 QCN_J01[1]	Query Cancel
	ACK	Cancel Acknowledgement

[1] The specification requires three components, while the sample data only contains the first two.

Table A–6 PIX v3 Supported Message Types

IHE Profile Name	HL7 Message Type	Action
ITI-44: Patient Identity Feed (v3)	PRPA_IN201301UV02	Add Patient
	PRPA_IN201302UV02	Update Patient
	PRPA_IN201304UV02	Merge Patients
	MCCI_IN000002UV01	Acknowledgement
ITI-45: PIX Query (v3)	PRPA_IN201309UV02	Query Message
	PRPA_IN201310UV02	Query Response
ITI-46: PIX Update Notification (v3)	PRPA_IN201302UV02	Update Notification
	MCCI_IN000002UV01	Acknowledgement

Table A–7 PDQ v3 Supported Message Types

IHE Profile Name	HL7 Message Type	Action
ITI-47: Patient Demographics Query (v3)	PRPA_IN201305UV02	Find Candidates Query
	PRPA_IN201306UV02	Query Response
	QUQI_IN000003UV01	Query Continue
	QUQI_IN000003UV01	Query Cancel
	MCCI_IN000002UV01	Cancel Acknowledgement

B

Adding IHE Profile Support to User-defined MPI Object Model

IHE Project comes with a pre-packaged MPI Project which has a fixed Object Model. This fixed MPI Object Model is not very useful in the real world. OHMPI 2.0 added capability to allow a user to add IHE Profile support to any user-defined MPI Object Model. However, before the 2.0.5 release, the process to add IHE Profile support to user-defined MPI Object Model requires a lot of manual re-packaging work after the IHE project is built. This issue has been addressed in the 2.0.5 release (#16719490).

In order to add IHE Profile support to user-defined MPI Object Model, user needs to do the following:

- Creating MPI Project on page 1
- Creating Java Mapper Project on page 2
- Creating All-in-one IHE Project on page 8

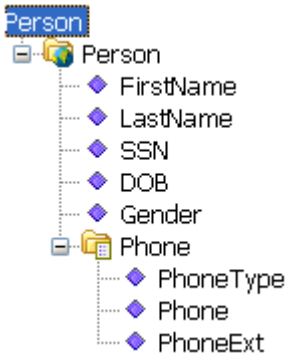
In the following example, we will use a simplified Person Object Model as the user-defined MPI Object Model to illustrate the process.

Creating MPI Project

1. Start from the Person Template.
2. Remove all attributes on the primary Person object except *FirstName*, *LastName*, *SSN*, *DOB*, and *Gender*.
3. Remove the *Alias* and *Address* child objects.

   Figure B–1 shows the Object Model of the user-defined MPI Project:

Figure B–1 Object Model



4. Generate Master Index Files.
5. Build the *Person* MPI Project.

Creating Java Mapper Project

The purpose of the Mapper Project is to provide two-way mapping between the user-defined MPI Object Model and the fixed IHE Object Model.

Figure B–2 displays a side-by-side comparison between the user-defined MPI Model and the fixed IHE Model:

Figure B–2 Comparison Between user-defined MPI Model and Fixed IHE Model

Table B–1 lists the field mapping.

Note: The *unformattedTelephoneNumber* in the fixed IHE Model only gets populated from HL7v3 feed. To support data coming from HL7 v2 feed, you need to check the other fields under the IHE Model's *PhoneNumber* object as well when doing the mapping. We skip that part here for simplicity.

Table B–1 Field Mapping

User-defined MPI Model	Fixed IHE Model
FirstName	givenName
LastName	familyName
SSN	SSN
DOB	dateOfBirth
Gender	sex
Phone.PhoneType	PhoneNumber.telecomUseCode
Phone.Phone	PhoneNumber.unformattedTelephoneNumber
Phone.PhoneExt	PhoneNumber.extension

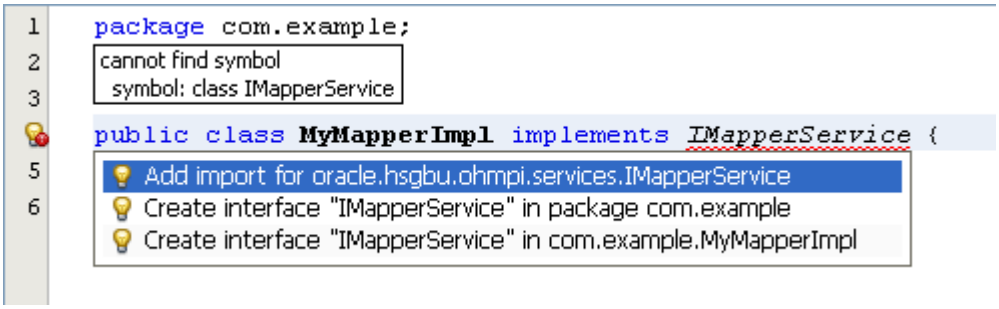
1. Create a new Java Class Library Project, say, *MyMapper.*
2. Add *index-core.jar* and *mpi-client-person.jar* from *Person/lib* into the library (Library -> Add Jar/Folder…).
3. Create a new IHE Project. For example, *MyIHE*.

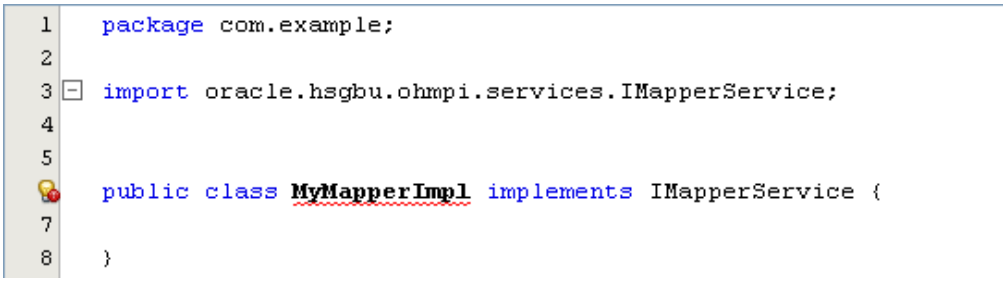
   An embedded MPI project (*MyIHE-mpi*) gets automatically generated.
4. Build the IHE Project.

   The *ihe-mpi-commons.jar* file is generated in the *MyIHE/lib* directory.
5. Add *ihe-mpi-commons.jar* from the *MyIHE/lib* directory into the library.
6. Create a new Java class. For example, *com.example.MyMapperImpl,* that implements *IMapperService*:

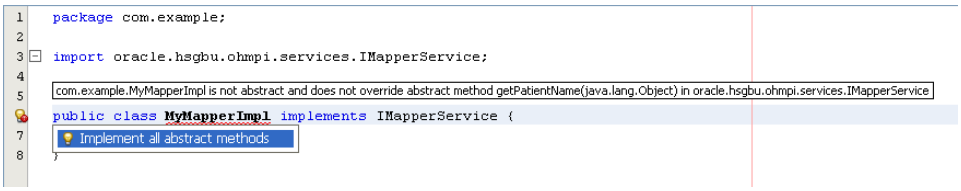
```
package com.example;public class MyMapperImpl implements IMapperService {}
```

7. Use IDE hint (or CTRL+SHIFT+I) to fix import.





8. Use IDE hint to provide a skeleton implementation of the mapper service.




```
1  package com.example;
2
3  import oracle.hsgbu.ohmpi.commons.PatientName;
4  import oracle.hsgbu.ohmpi.model.IHEPatientObject;
5  import oracle.hsgbu.ohmpi.services.IMapperService;
6
7
8  public class MyMapperImpl implements IMapperService {
9
10     public IHEPatientObject getIHEPatientObject(Object o) {
11         throw new UnsupportedOperationException("Not supported yet.");
12     }
13
14     public Object getMPIPatientObject(IHEPatientObject ihepo) {
15         throw new UnsupportedOperationException("Not supported yet.");
16     }
17
18     public PatientName getPatientName(Object o) {
19         throw new UnsupportedOperationException("Not supported yet.");
20     }
21
22 }
```

9. Take a look at the following Java Doc for the *oracle.hsgbu.ohmpi.services.IMapperService* Interface and understand the methods you need to implement:

oracle.hsgbu.ohmpi.services Interface IMapperService

`public interface IMapperService`
Interface to provide two-way mapping between the fixed IHE Patient Object Model and the user-defined MPI Patient Object Model.

Method Summary	
`IHEPatientObject`	**getIHEPatientObject**(`java.lang.Object mpiPatientObject`) Converts an MPI Patient Object instance to an IHE Patient Object instance.
`java.lang.Object`	**getMPIPatientObject**(`IHEPatientObjectihePatientObject`) Converts an IHE Patient Object instance to an MPI Patient Object instance.
`PatientName`	**getPatientName**(`java.lang.Object mpiPatientObject`) Convenience method to retrieve patient name information from an MPI Patient Object instance.

Method Details

getIHEPatientObject	
`IHEPatientObject getIHEPatientObject(java.lang.Object mpiPatientObject)`	Converts an MPI Patient Object instance to an IHE Patient Object instance.
Parameter	`mpiPatientObject`: A user-defined MPI Patient Object instance. This parameter is guaranteed to be an instance of MPI's `ObjectNode`. The service provider can safely cast it to the user-defined MPI Object Model's primary object in the implementation.

getIHEPatientObject	
Return	An IHE Patient Object instance

getMPIPatientObject	
`java.lang.Object getMPIPatientObject(IHEPatientObject ihePatientObject)`	Converts an IHE Patient Object instance to an MPI Patient Object instance.
Parameter	`ihePatientObject`: An IHE Patient Object instance.
Return	A user-defined MPI Patient Object instance. This has to be an instance of the user-defined MPI Object Model's primary object.

getPatientName	
`PatientName getPatientName(java.lang.Object mpiPatientObject)`	Convenience method to retrieve patient name information from an MPI Patient Object instance. This is to avoid a full-blown mapping from MPI Patient Object to IHE Patient Object under some circumstances.
Parameter	`mpiPatientObject`: A user-defined MPI patient object instance. This is guaranteed to be an instance of MPI's ObjectNode.
Return	Patient name

10. Provide your customized implementation of the mapper service. For this example, copy and paste the following content into *MyMapperImpl.java*:

```
package com.example;

import com.sun.mdm.index.objects.exception.ObjectException;
import com.sun.mdm.index.objects.person.PersonObject;
import com.sun.mdm.index.objects.person.PhoneObject;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.Date;
import java.util.logging.Logger;
import oracle.hsgbu.ohmpi.commons.PatientName;
import oracle.hsgbu.ohmpi.model.IHEPatientObject;
import oracle.hsgbu.ohmpi.model.IHEPhoneNumberObject;
import oracle.hsgbu.ohmpi.services.IMapperService;

/**
 * Customized mapper implementation to bridge the standard IHE Object Model and
customized MPI Object Model.
 */
public class MyMapperImpl implements IMapperService {

    private static final Logger logger =
Logger.getLogger(MyMapperImpl.class.getName());

    private DateFormat dateFormatter = new SimpleDateFormat("yyyyMMdd");

    public IHEPatientObject getIHEPatientObject(Object mpiPatientObject) {
        IHEPatientObject ihePO = new IHEPatientObject();

        try {
            PersonObject mpiPO = (PersonObject) mpiPatientObject;
```

```
            ihePO.setGivenName(mpiPO.getFirstName());
            ihePO.setFamilyName(mpiPO.getLastName());
            ihePO.setSex(mpiPO.getGender());
            ihePO.setSSN(mpiPO.getSSN());

            Date dob = mpiPO.getDOB();
            String dateOfBirthString = dateFormatter.format(dob);
            ihePO.setDateOfBirth(dateOfBirthString);

            if (mpiPO.getPhone() != null) {
                for (PhoneObject mpiPhone : (Collection<PhoneObject>)
mpiPO.getPhone()) {
                    IHEPhoneNumberObject ihePhone = new IHEPhoneNumberObject();
                    ihePhone.setExtension(mpiPhone.getPhoneExt());
                    ihePhone.setTelecomUseCode(mpiPhone.getPhoneType());

ihePhone.setUnformattedTelephoneNumber(mpiPhone.getPhone());
                    ihePO.addPhoneNumber(ihePhone);
                }
            }

        } catch (ObjectException ex) {
           logger.severe("Failed to convert MPI patient object to IHE patient
object.");
        }

        return ihePO;
    }

    public Object getMPIPatientObject(IHEPatientObject ihePO) {
        PersonObject mpiPO = null;
        try {
            mpiPO = new PersonObject();
            mpiPO.setFirstName(ihePO.getGivenName());
            mpiPO.setLastName(ihePO.getFamilyName());
            mpiPO.setGender(ihePO.getSex());
            mpiPO.setSSN(ihePO.getSSN());           // SSN is required in
user-defined Object Model

            if (ihePO.getDateOfBirth() != null) {
                try {
                    Date date = (Date)
dateFormatter.parse(ihePO.getDateOfBirth());
                    mpiPO.setDOB(date);
                } catch (ParseException ex) {
                    logger.warning("Failed to parse date string " +
ihePO.getDateOfBirth());
                }
            }

            if (ihePO.getPhoneNumbers() != null) {
                for (IHEPhoneNumberObject ihePhone : ihePO.getPhoneNumbers()) {
                    PhoneObject mpiPhone = new PhoneObject();

                    mpiPhone.setPhoneExt(ihePhone.getExtension());
                    mpiPhone.setPhoneType(ihePhone.getTelecomUseCode());

mpiPhone.setPhone(ihePhone.getUnformattedTelephoneNumber());

                    mpiPO.addPhone(mpiPhone);
```

```
                }
            }

        } catch (ObjectException ex) {
            logger.severe("Failed to convert IHE patient object to MPI patient
object.");
        }

        return mpiPO;
    }

    // Convenience method to avoid a full-blown mapping from MPI PO to IHE PO
    public PatientName getPatientName(Object mpiPatientObject) {
        try {
            PersonObject po = (PersonObject) mpiPatientObject;
            return new PatientName(
                    po.getLastName(), po.getFirstName(), null, null);
        } catch (ObjectException ex) {
            logger.severe("Failed to retrieve patient name from MPI patient
object.");
        }

        return null;
    }
}
```

11. Build the Java Mapper Project.

Creating All-in-one IHE Project

1. Update the following MPI Project references in the IHE project's *nbproject/project.properties* file:

   From

   - *mpi.project.name=MyIHE-mpi*
   - *project.MyIHE-mpi=MyIHE-mpi*
   - *reference.MyIHE-mpi.dist=${project.MyIHE-mpi}/dist/MyIHE-mpi.ear*

   To

   - *mpi.project.name=Person*
   - *project.MyIHE-mpi=../Person*
   - *reference.MyIHE-mpi.dist=${project.MyIHE-mpi}/dist/Person.ear*

   These three properties link the IHE Project to the MPI Project. The first one specifies the name of the MPI Project. The second one specifies the relative path of the MPI Project to the IHE Project. (Here, we are assuming the *Person* MPI Project is created at the same directory level as the IHE Project.) The third one specifies the build artifact of the MPI Project.

   Note: When you are updating the MPI Project references, do not change the property names. In the case of the third property, do not change the first part of the property value either.

2. Add a new property called *project.mapper* into the IHE project's *nbproject/project.properties* file. This is the relative path of the Java Mapper Project to

the IHE project. (Here, we are assuming the Java Mapper Project is created at the same directory level as the IHE Project.)

project.mapper=../MyMapper

3. Build the IHE project. This will package the user-defined MPI project instead of the embedded MPI project. It will also package the user-defined Java Mapper Project.
4. Under the application server's domain configuration directory, say, *domains/domain1/config* for GlassFish, or *user_projects/domains/base-domain/config* for WebLogic, create a new sub-directory called *ohmpi* and create an *ohmpi-ihe.properties* file under it. Add the following properties:
   - *# MPI Object Model's Primary Object Name*

     mpi/primary_object/name=Person
   - *# MPI Object Model's Primary Object Class Name*

     mpi/primary_object/class_name=com.sun.mdm.index.objects.person.PersonObject
   - *# IHE-MPI Mapper Service Implementation Class Name*

     service/mapper/class_name=com.example.MyMapperImpl

   Adjust the above property values accordingly if you are not following the names used in this document.
5. Create MPI and IHE database tables. For information, see Chapter 3, "Creating MPI and IHE Databases and Tables."
6. Configure application server resources. For information, see Chapter 4, "Configuring an Application Server."

   Note: Instead of using *Patient* as the name for JDBC and JMS Resource configuration, use the actual name of the Custom Object Model, say, *Person*. For example, *jdbc/PersonDataSource*, *jms/PersonOutBoundSender* and *jms/PersonTopic*. The same applies to the MIDM access.
7. Deploy the IHE project. For information, see Chapter 6, "Deploying the IHE Profiles Application."