Patient Identity Source

Architecture & API Documentation

Version 0.4.0

srrenly{at}us{dot}ibm{dot}com | Sondra R Renly

Contents

1. Introduction

Open Health Tools is an open source community with a vision of enabling a ubiquitous ecosystem where members of the Health and IT professions can collaborate to build interoperable systems that enable patients and their care providers to have access to vital and reliable medical information at the time and place it is needed. Open Health Tools will generate a vibrant active ecosystem involving software product and service companies, medical equipment companies, health care providers, insurance companies, government health service agencies, and standards organizations.

http://www.openhealthtools.org

The Open Health Tools IHE Profiles project participates in the committee meetings of relevant healthcare standards and interoperability organizations and brings that expertise to our open source community through the development and support of several key client side interoperability profile implementations. The IHE Profiles project will significantly lower the barriers towards the availability of interoperable healthcare systems.

https://iheprofiles.projects.openhealthtools.org

The Integrating the Healthcare Enterprise (IHE) is an initiative by healthcare professionals and industry to improve the way computer systems in healthcare share information. IHE promotes the coordinated use of established standards such as DICOM and HL7 to address specific clinical needs in support of optimal patient care. Systems developed in accordance with IHE communicate with one another better, are easier to implement, and enable care providers to use information more effectively.

http://www.ihe.net

The IHE Technical Frameworks are a resource for users, developers and implementers of healthcare imaging and information systems. They define specific implementations of established standards to achieve effective systems integration, facilitate appropriate sharing of medical information and support optimal patient care. They are expanded annually, after a period of public review, and maintained regularly by the IHE Technical Committees through the identification and correction of errata.

http://www.ihe.net/Technical_Framework/index.cfm

This document corresponds to the Open Health Tools IHE Profiles plugin implementation of the IHE ITI Technical Framework actor Patient Identity Source for the implementation of the ITI-8 Patient Identity Feed Transaction that was used successfully by several systems at the 2007 and 2008 IHE Connectathon. The code and documentation are released under the Eclipse Public License (EPL).

2. Getting Started

2.1 Platform Requirements

Verify that the following platform requirements are installed on your workstation, and if not follow the links provided to download and install.

Eclipse SDK 3.2 or higher	http://www.eclipse.org/downloads/
Java JDK 1.5.0 or higher	http://java.sun.com/javase/downloads/index.jsp

2.2 Source Files

Information on how to access the Open Health Tools SVN technology repository is found on the website:

https://iheprofiles.projects.openhealthtools.org/source/browse/iheprofiles/

Download:

- org.openhealthtools.ihe.utils
- org.openhealthtools.ihe.common.hl7v2.client
- org.openhealthtools.ihe.pix.source

For details regarding plugin contents, see the README.txt located in the resources/doc folder of each plugin.

2.3 Dependencies

The Patient Identity Source client has dependencies on other OHT plugins and external sources.

2.3.1 Other OHT Plugins

Patient Identity Source plugins are dependent on additional project plugins. You also need to check-out the following:

• org.eclipse.ohf.hl7v2.core org.eclipse.ohf.utilities org.apache.commons org.apache.axis org.xmlpull.v1	HL7v2 message object plugins and dependencies
• org.openhealthtools.ihe.common.mllp	Minimum Lower Level Protocol
• org.openhealthtools.ihe.atna.auditor org.openhealthtools.ihe.atna.nodeauth org.openhealthtools.ihe.atna.context	Auditing for messages sent and responses received
• org.apache.log4j	Debug, warning, and error logging

2.3.2 External Sources

Message defaults and message restrictions can be specified in an optional XML conformance profile. A sample ADT-A04 Register Outpatient (HL7v2.3.1) conformance profile is included with this plugin in the /resources/conf folder.

2.4 Resources

The following resources are recommended.

2.4.1 Other OHT plugin documentation

The following OHT plugin documents are related to the Patient Identity Source:

- OHT ATNA Audit Client

2.4.2 HL7 Standard 2.3.1

The Patient Identity Source references standards HL7 version 2.3.1.

http://www.hl7.org.

2.4.3 IHE ITI Technical Framework

Nine IHE IT Infrastructure Integration Profiles are specified as Final Text in the Version 2.0 ITI Technical Framework: Cross-Enterprise Document Sharing (XDS), Patient Identifier Cross-Referencing (PIX), Patient Demographics Query (PDQ), Audit trail and Node Authentication (ATNA), Consistent Time (CT), Enterprise User Authentication (EUA), Retrieve Information for Display (RID), Patient Synchronized Applications (PSA), and Personnel White Pages (PWP).

The IHE ITI Technical Framework can be found on the following website:
http://www.ihe.net/Technical_Framework/index.cfm#IT.

2.4.4 Newsgroup

Any unanswered technical questions may be posted to the OHT user discussion group.

3. API Documentation

The Patient Identity Source client supports three formats for input. The client will accept:

- a raw HL7 message (String)
- an HL7v2 message object (org.eclipse.ohf.hl7v2.core Message)
- an ITI-8 Patient Identity Feed message supporting the message construction of events:

    ADT_A01 – Admission of an inpatient into a facility
    ADT_A04 – Registration of an outpatient for a visit of the facility
    ADT_A05 – Pre-admission of an inpatient
    ADT_A08 – Update patient information
    ADT_A40 – Patient Merge – Internal ID

Examples for the three types of inputs are found in the org.openhealthtools.ihe.pix.source plugin.

org.openhealthtools.ihe.pix.source >

    src_tests > org.openhealthtools.ihe.pix.source.tests > HL7PixFeed.java
    src_tests > org.openhealthtools.ihe.pix.source.tests > MSGPixFeed.java
    src_tests > org.openhealthtools.ihe.pix.source.tests > ClientPixFeed.java

The files in src_tests use a TestConfiguration.java file for extracting the various file locations and MLLP connection parameters. Update this file with your settings before running the sample code.

A raw HL7 message string should be used as input when the originating application is fully capable of sending and receiving HL7 messages. In this case, the Patient Identity Source client is simply providing auditing, communication with the PIX server, and optional message verification. Server responses are returned to the caller as raw HL7 message strings. (HL7PixFeed)

A message object should be used as input when the originating application is directly using the OHF HL7v2 component which the Patient Identity Source client sits on top of. In this case, the application has taken full responsibility for message creation and reading the response. The Patient Identity Source client is simply providing conversion to raw HL7, auditing, communication with the PIX server, and optional message verification. Server responses are returned to the caller as HL7v2 message objects. (MSGPixFeed)

A ITI-8 Patient Identity Feed message should be used as input when the originating application has neither support for raw HL7 nor OHF HL7v2 message objects. The Patient Identity Source client provides a friendly interface to set and read message fields as well as auditing, communication with the PIX server, and optional message verification. Server responses are returned to the caller as PixSourceResponse objects. (ClientPixFeed)

ITI-8 Patient Identity Feed Message Classes

    PixMsgAdmitPatient
    PixMsgRegisterOutpatient
    PixMsgPreadmitInpatient
    PixMsgUpdatePatient
    PixMsgMergePatient

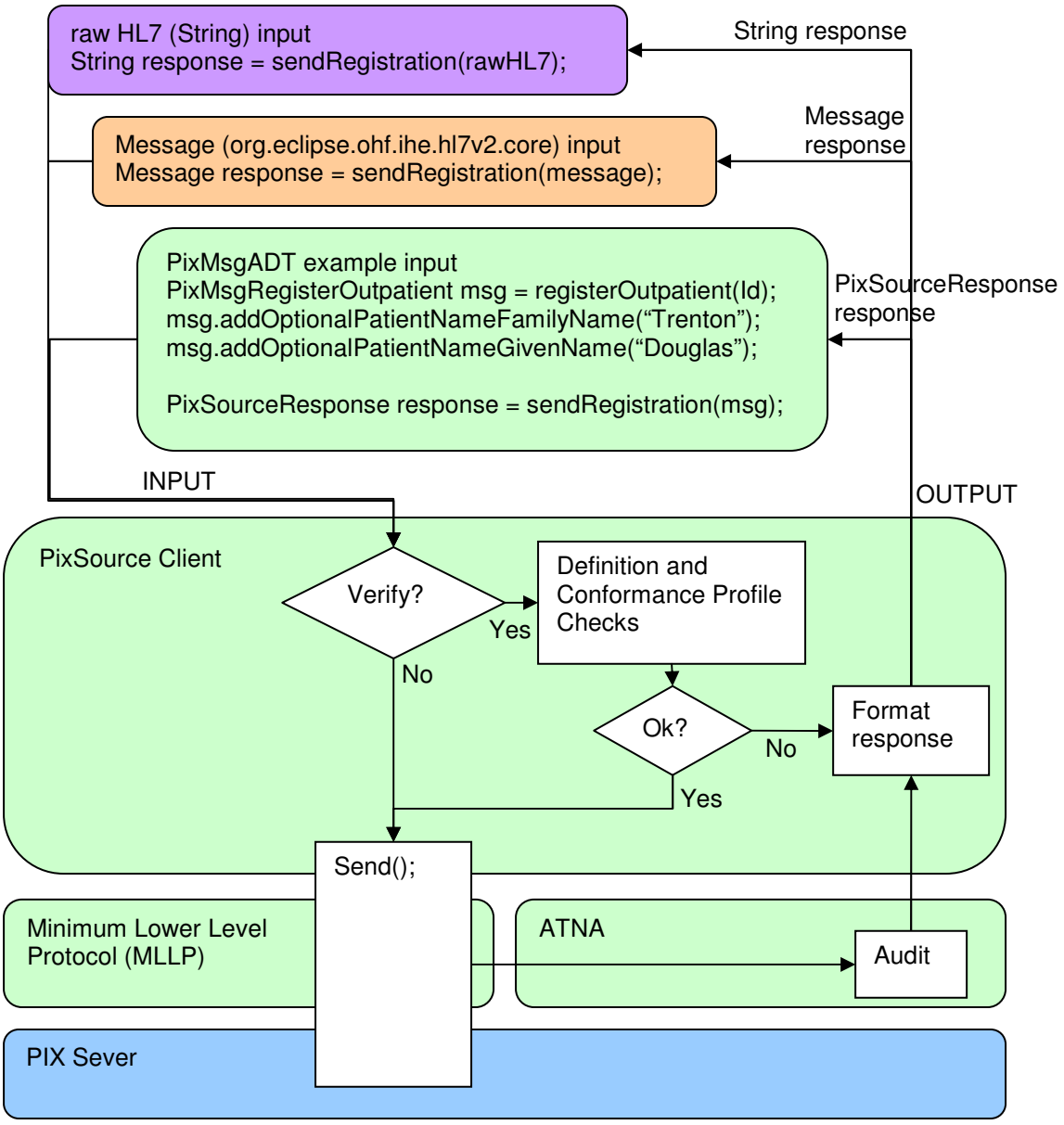
ITI-8 Patient Identity Feed Server Response Class

    PixSourceResponse

3.1 Use Case - ITI-8 Patient Identity Feed

The Patient Identity Source Client has one transaction. This use case demonstrates in step-by-step and with sample code the creation and use of the Patient Identity Source Client, including the three input options. It includes example client construction of the 5 possible event specific message objects as input but not the creation of raw HL7 or HL7v2 Message objects.

3.1.1 Flow of Execution



Create a Patient Identity Source object:

1. Construct ITI-8 Patient Identity Feed
2. Construct and associate MLLP (Minimum Lower Level Protocol) Destination
3. Configure auditing.
4. Override the maximum level of validation error allowed before message submission is halted. The levels of error are constants in the OHF HL7v2 CPValidator.java file. The default is to allow up to the warning level.

Create a tailored HL7v2 message object:

1. Create Patient Identity Source Message. Message field defaults are obtained first from the associated Conformance Profile.
2. Override default settings from conformance profile.
3. Add optional field values.
4. As not all fields have a corresponding method, use the generic method to set these additional values. Use method .setField(path, value).

The Patient Identity Source supports populating data from MSH, EVN, PID, MRG (if merge message), and PV1 segments. Information about the fields, components, and sub-components available in these segments is available in the HL7 Version 2.3.1 Standard document in Appendix B.

Send the message:

1. Send message

Read the response message:

1. Read response message fields.

3.1.2 Sample Code

Create a Patient Identity Source object:

1. Construct ITI-8 Patient Identity Feed

   There are two ways to construct the ITI-8 Patient Identity Feed client. At client creation, HL7 definitions are now automatically loaded for you from the HL7v2 toolkit. You can optionally provide an XML conformance profile for providing message defaults and additional message validation restrictions. The conformance profile can be added at a later time as well.

   In this sample code, TConfig refers to the TestConfiguration.java file mentioned in the beginning of this section. See this file for example formatting of these constants.

```
//pixFeed set-up
PixSource pixFeed = new PixSource();

//pixFeed set-up with conformance profile
InputStream cpStream = new FileInputStream(TConfig.CPROFILE_PATH);
PixSource pixFeed = new PixSource(cpStream);
```

```
cpStream.close();
```

2. Construct and associate MLLP (Minimum Lower Level Protocol) Destination

   Un-Secure Connection:

```
MLLPDestination mllp = new MLLPDestination(TConfig.MLLP_URI);
MLLPDestination.setUseATNA(true);
pixFeed.setMLLPDestination(mllp);
```

   Secure Connection:

```
MLLPDestination mllps = new MLLPDestination(TConfig.MLLPS_URI);
MLLPDestination.setUseATNA(true);
pixFeed.setMLLPDestination(mllps);

Properties props = new Properties();
props.setProperty(SecurityDomain.JAVAX_NET_SSL_KEYSTORE, "/x.jks”);
props.setProperty(SecurityDomain.JAVAX_NET_SSL_KEYSTORE_PASSWORD, "pswd");
props.setProperty(SecurityDomain.JAVAX_NET_SSL_TRUSTSTORE, "/y.jks");
props.setProperty(SecurityDomain.JAVAX_NET_SSL_TRUSTSTORE_PASSWORD, "pswd");
SecurityDomain domain = new SecurityDomain("domainXY", props);
ConfigurationManager.registerDefaultSecurityDomain(domain);
```

3. Configure auditing.

```
String auditUser = "Bob Smith";
PixSourceAuditor.getAuditor().getConfig().setAuditSourceId(auditUser);
PixSourceAuditor.getAuditor().getConfig().setAuditRepositoryUri(TConfig.ATNA_
URI);
```

4. Override the maximum level of validation error allowed before message submission is halted. The levels of error are constants in the OHF HL7v2 CPValidator.java file. The default is to allow up to the warning level.

```
ITEM_TYPE_INFORMATION = 1;
ITEM_TYPE_WARNING = 2;
ITEM_TYPE_ERROR = 3;
ITEM_TYPE_FATAL = 4;

pixFeed.setMaxVerifyEvent(CPValidator.ITEM_TYPE_INFORMATION);
```

Create a tailored HL7v2 message object:

1. Create Patient Identity Source Message.

```
//Event Option 1: ADT-01 Admit Inpatient
PixMsgAdmitInpatient msg = pixFeed.admitInpatient(patientId);

//Event Option 2: ADT-04 Register Outpatient
PixMsgRegisterOutpatient msg = pixFeed.registerOutpatient(patientId);

//Event Option 3: ADT-05 Preadmit Inpatient
PixMsgPreadmitInpatient msg = pixFeed.preadmitInpatient(patientId);

//Event Option 4: ADT-08 Update Patient
PixMsgUpdatePatient msg = pixFeed.updatePatient(patientId, class);

//Event Option 5: ADT-40 Merge Patient
PixMsgMergePatient msg = pixFeed.mergePatient(patientId, class, priorId);
```

2. Override default settings from conformance profile.

```
msg.changeDefaultCharacterSet("UNICODE");
```

3. Add optional field values.

```
msg.addOptionalPatientNameFamilyName("TRENTON");
msg.addOptionalPatientNameGivenName("DOUGLAS");
msg.addOptionalPatientAddressCity("SAN JOSE");
msg.addOptionalPatientAddressStateOrProvince("CA");
msg.addOptionalPatientAddressZipOrPostalCode("95120");
```

4. As not all fields have a corresponding method, use the generic method to set these additional values. Use method .setField(field, value).

```
msg.addOptionalPatientAddressStreetAddress("123 San Jose Drive");
msg.setField("PID-11-1", "123 San Jose Drive");
```

   For this example, the two statements are identical to show that the methods are equivalent.

Send the message:

1. Send message

```
//PixMsgADT message object

//Event Option 1: ADT-01 Admit Inpatient
PixSourceResponse response = pixFeed.sendAdmission(msg, isValidateOn);

//Event Option 2: ADT-04 Register Outpatient
PixSourceResponse response = pixFeed.sendRegistration(msg, isValidateOn);

//Event Option 3: ADT-05 Preadmit Inpatient
```

```
PixSourceResponse response = pixFeed.sendPreAdmission(msg, isValidateOn);

//Event Option 4: ADT-08 Update Patient
PixSourceResponse response = pixFeed.sendUpdate(msg, isValidateOn);

//Event Option 5: ADT-40 Merge Patient
PixSourceResponse response = pixFeed.sendMerge(msg, isValidateOn);
```

Read the response message:

1. Read response

```
//HL7v2 message object
msg.getElement("MSA-1").getAsString();    //message AckCode

//PixSourceResponse object
response.getResponseAckCode(isExpandCodeToString);
response.getControlId();
response.getErrorCodeAndLocation();
```

4. Security

4.1 Node Authentication

Transport Layer Security Protocol (TLS) is supported by creating a secure MLLP connection. Information required to instantiate a secure connection to one of the IBM Public Servers is available in the sample code configuration file.

4.2 Auditing

Auditing to an Audit Record Repository is automatically enabled through the ATNA Agent. The Patient Identity Source automatically generates the following audit messages:

- EventID 110100 - Actor Start audit message (EventTypeCode 110120)
- EventID 110106 - Export audit message
- EventID 110100 - Actor Stop audit message (EventTypeCode 110121)

Configuration

There are two types of configuration in this release.

4.3 Conformance Profile

Create message default field values, such as message header fields, can now be read from the conformance profile field ConstantValue attribute. This is now supported at all levels: field, component, and sub-component.

Field example:

```
<Field Name="MSH-1 Field Separator" Usage="R" Min="1" Max="1" Datatype="ST"
Length="1" ItemNo="00001" ConstantValue="|"></Field>
```

Component example (in this case only the namespaceId is defaulted):

```
<Field Name="MSH-3 Sending Application" Usage="R" Min="0" Max="1" Datatype="HD"
Length="227" Table="0361" ItemNo="00003">

      <Component Name="MSH-3-1 sending application: namespace ID" Usage="R"
      Datatype="IS" ConstantValue="OHTConsumer1"></Component>

      <Component Name="MSH-3-2 sending application: universal ID" Usage="O"
      Datatype="ST"></Component>

      <Component Name="MSH-3-3 sending application: universal ID type" Usage="O"
      Datatype="ID"></Component>

</Field>
```

Sub-component example (specifies a limit of 5 records):

```
<Field Name="RCP-2 Quantity Limited Request" Usage="O" Min="0" Max="1"
Datatype="CQ" Length="10" Table="0126" ItemNo="00031">

      <Component Name="RCP-2-1 quantity limited request: quantity" Usage="O"
      Datatype="NM" ConstantValue="5"></Component>

      <Component Name="RCP-2-2 quantity limited request: units" Usage="O"
      Datatype="CE">

            <SubComponent Name="RCP-2-2-1 quantity limited request: units:
            identifer" Usage="O" Datatype="ST"
            ConstantValue="RD"></SubComponent>

            <SubComponent Name="RCP-2-2-2 quantity limited request: units: text"
            Usage="O" Datatype="ST"></SubComponent>

            <SubComponent Name="RCP-2-2-3 quantity limited request: units: name
            of coding system" Usage="O" Datatype="ID"></SubComponent>

            <SubComponent Name="RCP-2-2-4 quantity limited request: units:
            alternate identifier" Usage="O" Datatype="ST"></SubComponent>

            <SubComponent Name="RCP-2-2-5 quantity limited request: units:
            alternate text" Usage="O" Datatype="ST"></SubComponent>

      </Component>

</Field>
```

4.4 Test Configuration

The files in src_tests now use a TestConfiguration.java file for extracting the various file locations and MLLP connection parameters. Update this file with your settings before running the sample code. Here are the fields that are configured in this file:

```
//basics
public static final String DATA_PATH
public static final String LOG4J_PATH

//HL7PixQuery - run from file
public static final String HL7FILE_PATH

//Conformance profile for second level HL7 verification and defaults
public static final String CPROFILE_PATH

//MLLP Connectivity:
public static URI MLLP_URI
public static URI MLLPS_URI

//TLS: Secure connection parameters
public static final String MLLP_KEYSTORE_NAME
public static final String MLLP_KEYSTORE_PASSWORD
public static final String MLLP_TRUSTSTORE_NAME
public static final String MLLP_TRUSTSTORE_PASSWORD
```

5. Debugging Recommendations

Log statements have been entered throughout the Patient Identity Source plugin source code for assistance in monitoring the progress of the running client. To enable logging, there is a Log4j configuration.

An example log4j configuration is in the file/resources/conf/pixsource_log4j.xml. The default configuration writes to a log file in the folder /resources/log.

6. IHE Connectathon MESA Tests

For current information, please refer to the website:

6.1 Plugin Testing

The OHT Patient Identity Source plugin completed testing with the following junits and test scripts.

org.openhealthtools.ihe.pix.source > src_tests > org.openhealthtools.ihe.pix.source.test.mesa >

MESA10512Test.java	Test10512a – PIX Patient Feed: A04 Test A Test10512b – PIX Patient Feed: A04 Test B Test10512c – PIX Patient Feed: A04 Test C
MESA2009.txt	Sample script for starting the mesa server and running all tests.

6.2 Bridge Testing

The OHT Bridge completed testing with the following junits and test scripts.

org.openhealthtools.bridge.ws > src_tests > org.openhealthtools.bridge.ws.tests.mesa >

TestMESA10512a.java	Test10512a – PIX Patient Feed: A04 Test A
TestMESA10512b.java	Test10512b – PIX Patient Feed: A04 Test B
TestMESA10512c.java	Test10512c – PIX Patient Feed: A04 Test C

org.openhealthtools.ihe.pix.source > src_tests > org.openhealthtools.ihe.pix.source.test.mesa >

MESA2009.txt	Sample script for starting the mesa server and running all tests.

7. IHE Connectathon Tests

7.1 Plugin Testing

The OHT Patient Identity Source plugin completed testing with the following junits.

org.openhealthtools.ihe.pix.source > src_tests > org.openhealthtools.ihe.pix.source.test.connectathon >

PIXConnectathonTest.java	2.1 PIX_Seed_Mgr 2.2 PIX_FEED
ConnectathonPDQLoadTest.java	2.1 PDQ_Load The initial patient load for PDQ requires use of the Patient Identity Source plugin.