Patient Identity Source

Architecture & API Documentation

Version 0.1.0

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

Contents

1. Introduction

The Eclipse Foundation is a not-for-profit corporation formed to advance the creation, evolution, promotion, and support of the Eclipse Platform and to cultivate both an open source community and an ecosystem of complementary products, capabilities, and services. Eclipse is an open source community whose projects are focused on providing an extensible development platform and application frameworks for building software.

www.eclipse.org

The Eclipse Open Healthcare Framework (Eclipse OHF) is a project within Eclipse formed for the purpose of expediting healthcare informatics technology. The project is composed of extensible frameworks and tools which emphasize the use of existing and emerging standards in order to encourage interoperable open source infrastructure, thereby lowering integration barriers.

www.eclipse.org/ohf

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.

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 documentation addresses the beta release of the Eclipse OHF plugin implementation of the IHE ITI Technical Framework actor Patient Identity Source for the implementation of the ITI-8 Patient Identity Feed Transaction.

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	http://www.eclipse.org/downloads/
Java JDK 1.4.2	http://java.sun.com/javase/downloads/index.jsp

2.2 Source Files

Information on how to access the Eclipse CVS technology repository is found on the eclipse wiki:

http://wiki.eclipse.org/index.php/CVS_Howto

Download from dev.eclipse.org/technology/org.eclipse.ohf/plugins:

- org.eclipse.ohf.ihe.common.hl7v2.client
- org.eclipse.ohf.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 OHF plugins and external sources.

2.3.1 Other OHF Plugins

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

• org.eclipse.ohf.hl7v2.core org.eclipse.ohf.utilities org.apache.axis org.apache.commons org.xmlpull.v1	HL7v2 message object plugins and dependencies
• org.eclipse.ohf.ihe.common.mllp	Minimum Lower Level Protocol
• org.eclipse.ohf.ihe.atna.audit	Auditing for messages sent and responses received
• org.eclipse.ohf.ihe.common.hl7v2	HL7v2 segment and field definitions (temporary)
• org.apache.log4j	Debug, warning, and error logging

2.3.2 External Sources

For the purpose of message object creation and verification, either a licensed copy of the HL7 access database (hl7_58.mdb) or a free javaStream file (hl7Definitions.javaStream) is necessary. You can obtain the free javaStream file from:

org.eclipse.ohf.hl7v2.ui > Resources > hl7Definitions.javaStream.

If you would like to obtain a copy of the HL7 access database file, refer to http://www.hl7.org.

To complete message verification, an XML conformance profile is necessary. 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 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.

2.4.2 HL7 Standard 2.3.1

The Patient Identity Source references standards HL7 version 2.3.1.

http://www.hl7.org.

2.4.3 Newsgroup

Any unanswered technical questions may be posted to Eclipse OHF newsgroup. The newsgroup is located at news://news.eclipse.org/eclipse.technology.ohf.

You can request a password at: http://www.eclipse.org/newsgroups/main.html

3. API Documentation

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

- a raw HL7 message
- an HL7v2 message object
- an ITI-8 Patient Identity Feed message supporting the HL7v2 message construction of:

  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.eclipse.ohf.ihe.pix.source plugin.

org.eclipse.ohf.ihe.pix.source > src_tests > org.eclipse.ohf.ihe.pix.source.tests > HL7PixFeed.java
org.eclipse.ohf.ihe.pix.source > src_tests > org.eclipse.ohf.ihe.pix.source.tests > MSGPixFeed.java
org.eclipse.ohf.ihe.pix.source > src_tests > org.eclipse.ohf.ihe.pix.source.tests > OtherPixFeed.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 optional message verification, auditing, and communicating with the PIX server. Server responses are returned to the caller as raw HL7v2 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, optional message verification, auditing, and communicating with the PIX server. 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 rawHL7 nor message objects. The Patient Identity Source client provides a friendly interface to set and read message fields as well as optional message verification, auditing, and communicating with the PIX server. (OtherPixFeed)

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.

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. Enable auditing/logging.
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 if no raw HL7 or message object:

1. Create Patient Identity Source Message. Message field defaults are obtained first from the associated Conformance Profile. Required fields not found there default to settings for the IBM Dallas Server. (http://ibmod235.dal-ebis.ihost.com:9080/IBMIHII/serverInfoOHF.htm)
2. Change default settings.
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(field, 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

```
//pixFeed set-up: no supporting files
PixSource pixFeed = new PixSource();
```

```
//pixFeed set-up: supporting access database & conformance profile
String msAccessFile = TConfig.ACCESS_DATABASE_PATH;
InputStream cpaStream = new FileInputStream(TConfig.CPROFILE_PATH);
PixSource pixFeed = new PixSource(msAccessFile, cpaStream);

//pixFeed set-up: javaStream setup example
Stream javaStream = new Stream(new File(TConfig.SERIALISED_PATH));
InputStream cpStream = new FileInputStream(TConfig.CPROFILE_PATH);
PixSource pixFeed = new PixSource(javaStream, cpStream);
```

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

```
MLLPDestination mllp =
new MLLPDestination(TConfig.MLLP_HOST, TConfig.MLLP_PORT);
pixFeed.setMLLPDestination(mllp);
```

3. Enable auditing/logging.

```
pixFeed.setDoAudit(true);
```

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 if no raw HL7 or message object:

1. Create Patient Identity Source Message.

```
//ADT-01 Admit Inpatient
PixMsgAdmitInpatient msg = pixFeed.admitInpatient("[patientId]");

//ADT-04 Register Outpatient
PixMsgRegisterOutpatient msg = pixFeed.registerOutpatient("[patientId]");

//ADT-05 Preadmit Inpatient
PixMsgPreadmitInpatient msg = pixFeed.preadmitInpatient("[patientId]");


//ADT-08 Update Patient
PixMsgUpdatePatient msg = pixFeed.updatePatient("[patientId]", "[class]");

//ADT-40 Merge Patient
PixMsgMergePatient msg = pixFeed.mergePatient("[patientId]", "[class]",
"[priorId]");
```

2. Change default settings.

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

3. Add optional field values.

   ```
   msg.addOptionalPatientNameFamilyName("RENLY");
   msg.addOptionalPatientNameGivenName("HALLEY");
   msg.addOptionalPatientAddressCity("SAN JOSE");
   msg.addOptionalPatientAddressStateOrProvince("CA");
   msg.addOptionalPatientAddressZipOrPostalCode("95123");
   ```

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

   ```
   String response = pixFeed.sendHL7(rawHL7, true, auditUser);
   Message response = pixFeed.sendMsg(msg, true, auditUser);

   PixSourceResponse response;

   //ADT-01 Admit Inpatient
   response = pixFeed.sendAdmission(msg, true, auditUser);
   //ADT-04 Register Outpatient
   response = pixFeed.sendRegistration(msg, true, auditUser);
   //ADT-05 Preadmit Inpatient
   response = pixFeed.sendPreAdmission(msg, true, auditUser);
   //ADT-08 Update Patient
   response = pixFeed.sendUpdate(msg, true, auditUser);
   //ADT-40 Merge Patient
   response = pixFeed.sendMerge(msg, true, auditUser);
   ```

Read the response message:

1. Read response

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

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

4. Security

[TODO]

4.1 Node Authentication

[TODO]

4.2 Auditing

Currently, ATNA auditing is not yet available. For now the auditing is done simultaneously with logging and to the same log as the Log4j logger. Auditing is enabled with the doAudit boolean variable used within the Patient Identity Source client.

```
boolean doAudit = true;

PixSource pixFeed = new PixSource();
pixFeed.setDoAudit(doAudit);
```

5. Configuration

There are two types of configuration in this release.

Create message default field values, such as message header fields, can now be read from the conformance profile field ConstantValue attribute. Currently this is only for single component fields. Overloading the string with component or sub-component delimited data is not supported.

```
<Field Name="MSH-1 Field Separator" … ConstantValue="|">
<Field Name="MSH-2 Encoding Characters" … ConstantValue="^~\&amp;">
<Field Name="MSH-3 Sending Application" … ConstantValue="OTHER_KIOSK">
<Field Name="MSH-4 Sending Facility" … ConstantValue="HIMSSSANDIEGO">
```

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:

```
DATA_PATH = "C:\\workspace";
LOG4J_PATH = DATA_PATH + "pixsource_log4j.xml";
HAS_ACCESSFILE = true;
ACCESS_DATABASE_PATH = DATA_PATH + "hl7_58.mdb";
HAS_JAVASTREAM = true;
SERIALISED_PATH = DATA_PATH + "hl7Definitions.javaStream";
CPROFILE_PATH = DATA_PATH + "ADT-A04(register outpatient).XML";
MLLP_HOST = "ibmod235.dal-ebis.ihost.com";
MLLP_PORT = 3600;
```

6. 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 file and boolean enabling variable for both the PixSource client and MLLPDestination communication layer.

The Log4j configuration needed is in the file /resources/conf/pixsource_log4j.xml. The default configuration creates the log file in the folder /resources/log.

The variable configuration is supported at both the Client and MLLP communication layer. Each must be specified independently as shown below:

```
boolean doAudit = false;

PixSource pixFeed = new PixSource();
pixFeed.setDoAudit(doAudit);

MLLPDestination mllp =
new MLLPDestination(TConfig.MLLP_HOST, TConfig.MLLP_PORT);
mllp.setDoAudit(doAudit);
```

7. Release Fixes and Enhancements

The primary enhancement of this release is the addition of the Eclipse OHF HL7v2 message verification functionality. HL7v2 message verification relies on a two step process; a definition file and an XML conformance profile. The definition file comes from HL7 while the XML conformance profile is configurable to allow site specific limitations.

HL7v2 message verification is currently an incomplete implementation. Monitor Eclipse Bugzilla for the HL7v2 OHF component to determine when it will be advantageous to enable message verification.

7.1 Release Fixes

1. Implemented the PixMsgADT addOptionalPatientId method. Before, this method's code had been commented out. Now, the code has been uncommented and it correctly sets the 4 components of the patient Id field.
2. Corrected the PixSourceResponse method getResponseAck, changing it to getResponseAckCode.
3. Corrected several of the PID segment get method names for consistency. For example, getSex was changed to getPatientSex.
4. Corrected variable names containing "ID", changing them to "Id" for consistency. For example, getControlID was changed to getControlId.
5. Corrected variable names containing "_", changing them to Java standard for consistency. For example, patient_class was changed to patientClass.
6. Corrected an index out of bounds error that occurred when accessing the MSH Segment getSendingFacility method for a server response message.

7.2 Release Enhancements

1. Implementations using only rawHL7 strings with no intermediate verification are now able to instantiate the Patient Identity Source without the HL7 definition file. This dependency has been removed. If no HL7 definition file is provided and verification is requested in the sendHL7 method, a log message is generated and the message sent without verification.

   ```
   PixSource pixFeed = new PixSource();
   pixFeed.setMLLPDestination(createMLLP());
   String message = createMessageA04();
   String response = pixFeed.sendHL7(message, false, auditUser);
   readReturn(response);
   ```

2. Added Eclipse OHF HL7v2 message verification functionality. HL7v2 message verification relies on a two step process, a definition file and an XML conformance profile. The definition file comes from HL7 and can be either the HL7 access database file (licensed through HL7) or the javaStream file that is now provided free with the HL7v2 component. The XML conformance profile is configurable to allow site specific limitations. A sample ADT_A04 conformance profile is included in the Patient Identity Source component.

3. HL7 access database file example:
```
String msAccessFile = TConfig.ACCESS_DATABASE_PATH;
InputStream cpaStream = new FileInputStream(TConfig.CPROFILE_PATH);
PixSource pixFeed = new PixSource(msAccessFile, cpaStream);
String response = pixFeed.sendHL7(message, true, auditUser);
```

   javaStream file example:
```
Stream javaStream = new Stream(new File(TConfig.SERIALISED_PATH));
InputStream cpStream = new FileInputStream(TConfig.CPROFILE_PATH);
PixSource pixFeed = new PixSource(javaStream, cpStream);
String response = pixFeed.sendHL7(message, true, auditUser);
```

4. Create message default field values, such as message header fields, can now be read from the conformance profile field ConstantValue attribute. Currently this is only for single component fields. Overloading the string with component or sub-component delimited data is not supported.
```
<Field Name="MSH-1 Field Separator" … ConstantValue="|">
<Field Name="MSH-2 Encoding Characters" … ConstantValue="^~\&amp;">
<Field Name="MSH-3 Sending Application" … ConstantValue="OTHER_KIOSK">
<Field Name="MSH-4 Sending Facility" … ConstantValue="HIMSSSANDIEGO">
```

5. Stop the submission of a message based on the maximum level of allowed errors reported by HL7v2 message verification. The default is to only send a message that has received information or warning errors. The four levels of error include:
```
ITEM_TYPE_INFORMATION = 1;
ITEM_TYPE_WARNING = 2;
ITEM_TYPE_ERROR = 3;
ITEM_TYPE_FATAL = 4;

PixSource pixFeed = new PixSource(javaStream, cpStream);
pixFeed.setMaxVerifyEvent(Validator.ITEM_TYPE_INFORMATION);  //block warnings
String response = pixFeed.sendHL7(message, true, auditUser);
```

6. Audit user is now entered at the transaction level rather than the client level.

   Before.
```
PixSource pixFeed = new PixSource(msAccessFile, cpaStream);
pixFeed.setAuditUser(auditUser);
```

   Now,
```
PixSource pixFeed = new PixSource(msAccessFile, cpaStream);
PixMsgRegisterOutpatient msg = createMessageA04(pixFeed);
PixSourceResponse response =
pixFeed.sendRegistration(msg, true, auditUser);
```

7. 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:
```
DATA_PATH = "C:\\workspace";
LOG4J_PATH = DATA_PATH + "pixsource_log4j.xml";
HAS_ACCESSFILE = true;
ACCESS_DATABASE_PATH = DATA_PATH + "hl7_58.mdb";
```

```
HAS_JAVASTREAM = true;
SERIALISED_PATH = DATA_PATH + "hl7Definitions.javaStream";
CPROFILE_PATH = DATA_PATH + "ADT-A04(register outpatient).XML";
MLLP_HOST = "ibmod235.dal-ebis.ihost.com";
MLLP_PORT = 3600;
```

8. The MLLP constructor now assumes the start and end characters for MLLP. The buffer size is also automatically set to 4096, but configurable using the setBufferSize() method in TCPPort.java.

   Before,
   ```
   MLLPDestination mllp =
   new MLLPDestination(host, port, startChar, endChar, buffersize);
   ```

   Now,
   ```
   MLLPDestination mllp =
   new MLLPDestination(TConfig.MLLP_HOST, TConfig.MLLP_PORT);
   ```

9. The HL7v2 definition file is no longer fixed to a folder within the Client plugin. It is now a parameter of the Client constructor.
10. Javadoc and readme files are now released with the plugin.

7.3 Available Workarounds

Eclipse Bugzilla 150336 JRE hard coded in org.eclipse.ohf.utilities:

After downloading the org.eclipse.ohf.utilities plugin, go to the build path configuration Libraries tab and remove the inconsistent JRE name. Replace it with the JRE_LIB variable or your specific workspace library.

Eclipse Bugzilla 151023 javaStream validation failing:

After downloading the org.eclipse.ohf.hl7v2.core plugin, go to org.eclipse.ohf.hl7v2.core.definitions.model, NamedDefn.java and change the method getName() as shown. Using the Path variable to abstract name gets around this error today.

```
public String getName() {
      int lastSep = getPath().lastIndexOf(PATH_SEPARATOR);
      return getPath().substring(lastSep+1);
      //return name;
}
```