Patient Demographics Consumer 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 Demographics Consumer for the implementation of the ITI-21 Patient Demographics Query 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.pdq.consumer

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

2.3 Dependencies

The Patient Demographics Consumer has dependencies on both other OHF plugins and external sources.

2.3.1 Other OHF Plugins

Patient Demographics Consumer 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, 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 Q22 Find Candidates (HL7v2.5) 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 2.5

The Patient Demographics Consumer references standards HL7 version 2.5.

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 Demographics Consumer client supports three formats for input. The client will accept:

- a raw HL7 message
- an HL7v2 message object
- an ITI-21 Patient Demographics Query message supporting the construction of:

    QBP^Q22 Patient Demographics Query

Examples for the three types of inputs are found in the org.eclipse.ohf.ihe.pdq.consumer plugin.

org.eclipse.ohf.ihe.pdq.consumer > src_tests > org.eclipse.ohf.ihe.pdq.consumer.tests > HL7PdqQuery.java
org.eclipse.ohf.ihe.pdq.consumer > src_tests > org.eclipse.ohf.ihe.pdq.consumer.tests > MSGPdqQuery.java
org.eclipse.ohf.ihe.pdq.consumer > src_tests > org.eclipse.ohf.ihe.pdq.consumer.tests > OtherPdqQuery.java

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 Demographics Consumer 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. (HL7PdqQuery)

A message object should be used as input when the originating application is directly using the OHF HL7v2 component which the Patient Demographics Consumer client sits on top of. In this case, the application has taken full responsibility for message creation and reading the response. The Patient Demographics Consumer 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. (MSGPdqQuery)

A ITI-21 Patient Demographics Query message should be used as input when the originating application has neither support for rawHL7 nor message objects. The Patient Demographics Consumer client provides a friendly interface to set and read message fields as well as optional message verification, auditing, and communicating with the PIX server. (OtherPdqQuery)

ITI-21 Patient Demographics Query Message Class

    PdqConsumerQuery

ITI-21 Patient Demographics Query Server Response Class

    PdqConsumerResponse

3.1 Use Case - ITI-21 Patient Demographics Query

The Patient Demographics Consumer has one transaction.

3.1.1 Flow of Execution

Create a Patient Demographics Consumer object:

1. Construct ITI-21 Patient Demographics Query
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 Demographics Consumer 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 query values.
4. As not all fields have a corresponding method, use the generic method to set these additional values. Use method .addOptionalDemographicSearch(alias, data).

The Patient Demographics Consumer supports querying data from PID and PD1 segments. Information about the fields, components, and sub-components available in these segments is available in the HL7 Version 2.5 Standard document in Chapter 3 Section 3.4 Message Segments.

Send the message:

1. Send message

Read the response message:

1. Read response message fields.

3.1.2 Sample Code

Create a Patient Demographics Consumer Query:

1. Construct ITI-21 Patient Demographics Query

```
//pdqQuery set-up: no supporting files
pdqQuery = new PdqConsumer();

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

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

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

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

3. Enable auditing/logging.

```
pdqQuery.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;

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

Create a tailored HL7v2 message object if no raw HL7 or message object:

1. Create Patient Demographics Consumer Message.

```
PdqConsumerQuery msg = pdqQuery.createQuery("[patientID]");
```

2. Change default settings.

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

3. Add optional query values.

```
msg.addOptionalQuerySex("F");
```

4. As not all fields have a corresponding method, use the generic method to set these additional values. Use method .addOptionalDemographicSearch(alias, data).

```
msg.addOptionalQuerySex("F");
msg.addOptionalDemographicSearch("PID-8", "F");
```

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

Send the message:

1. Send message

```
String response = pdqQuery.sendHL7(msg, true, auditUser);
Message response = ppdQuery.sendMsg(msg, true, auditUser);
PdqConsumerResponse response = pdqQuery.sendQuery(msg, true, auditUser);
```

Read the response message:

1. Read response

```
//HL7v2 message object
msg.getElement("MSA-1").getAsString();    //message AckCode
msg.getElement("QAK-2").getAsString();    //message QueryStatus

//PdqConsumerResponse object
response.getResponseAckCode(true);
```

```
response.getQueryStatus(true);
response.getPatientCount();
for (int i=1; i <= response.getPatientCount(); i++) {
      response.getPatientIdentifierIdNumber(i);
      response.getPatientNameFamilyName(i));
}
```

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 Demographics Consumer client.

```
boolean doAudit = true;

PdqConsumer pdqQuery = new PdqConsumer();
pdqQuery.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="OHFConsumer1">
<Field Name="MSH-4 Sending Facility" … ConstantValue="OHFFacility1">
```

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 + "pdqconsumer_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 + "QBP-Q22(find candidates).XML";
MLLP_HOST = "ibmod235.dal-ebis.ihost.com";
MLLP_PORT = 3600;
```

6. Debugging Recommendations

Log statements have been entered throughout the Patient Demographics Consumer 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 PDQ Consumer client and MLLPDestination communication layer.

The Log4j configuration needed is in the file /resources/conf/pdqconsumer_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;

PdqConsumer pdqQuery = new PdqConsumer();
pdqQuery.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. Corrected the PdqConsumerResponse method getPatientAddressZipOrPostalCode method. Before, this method was looking at the wrong HL7 field. Now, the code is correctly returning the zip field.
2. Corrected the PdqConsumerResponse 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 Demographics Consumer 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.

```
PdqConsumer pdqQuery = new PdqConsumer();
pdqQuery.setMLLPDestination(createMLLP());
String message = createMessage();
String response = pdqQuery.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 Q22 conformance profile is included in the Patient Demographics Consumer component plugin.

HL7 access database file example:

```
String msAccessFile = TConfig.ACCESS_DATABASE_PATH;
InputStream cpaStream = new FileInputStream(TConfig.CPROFILE_PATH);
PdqConsumer pdqQuery = new PdqConsumer(msAccessFile, cpaStream);
String response = pdqQuery.sendHL7(message, true, auditUser);
```

javaStream file example:

```
Stream javaStream = new Stream(new File(TConfig.SERIALISED_PATH));
InputStream cpStream = new FileInputStream(TConfig.CPROFILE_PATH);
PdqConsumer pdqQuery = new PdqConsumer(javaStream, cpStream);
String response = pdqQuery.sendHL7(message, true, auditUser);
```

3. 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">
```

4. 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;

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

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

Before.

```
PdqConsumer pdqQuery = new PdqConsumer(msAccessFile, cpaStream);
pdqQuery.setAuditUser(auditUser);
```

Now,

```
PdqConsumer pdqQuery = new PdqConsumer(msAccessFile, cpaStream);
PdqConsumerQuery msg = createMessage(pdqQuery);
PdqConsumerResponse response =
pdqQuery.sendQuery(msg, true, auditUser);
```

6. 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 + "pdqconsumer_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 + "QBP-Q22(find candidates).XML";
MLLP_HOST = "ibmod235.dal-ebis.ihost.com";
MLLP_PORT = 3600;
```

7. 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);
   ```

8. The HL7v2 definition file is no longer fixed to a folder within the Client plugin. It is now a parameter of the Client constructor.
9. 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;
}
```