Java CAPS Quick Note 002
JCA Transform HL7 Delimited to Custom XML

Michael.Czapski@sun.com
28th of February 2008

This Quick Note discusses a solution to the use case provided by Marcus Davies.

I am trying to read HL7 from JMS (preferably stcms) and populate an outbound XML data structure (different to the XML generated by the decoder).

I have been thinking of doing one of the following […]:

1. Use a Concrete JMS WS using the HL7 encoders to unmarshal the HL7 and use JAXB to populate the outbound XML. Unfortunately this does not appear to connect to the stcms queue as I can not see any receivers
2. Use a JCA MDB to read from the stcms JMS queue - this works but I don't think I can use the HL7 encoder like this
3. Use and MDB to read from JMS, manually unmarshal the HL7 and use JAXB to populate the data structure

Ideally I would like to use the HL7 encoders. Do you think the first approach should work?

Number 1 will not work as at end of February 2009 because the JMS BC does not properly decode the HL7 delimited message. This is a know issue. I don't know what the status of this is. The only BCs that know how to deal with HL7 delimited, that I know of, are the File BC and the HL7 BC.

Number 2 should work. I did not personally try it. You can invoke an encoder library from Java. Have a look at http://wiki.open-esb.java.net/Wiki.jsp?page=UseEncodersInJavaSE.

Number 3 should work but it will be very laborious.

I have a Number 4, which uses a HL7 OTD and a custom XSD-based OTD in a JCA EJB. You may or may not like it but it's the best thing to do if you can not use BPEL 2.0 to do the mapping and you don't want to build a repository-based solution (which would be the best for your case anyway).

The solution involves the use of:

1. HL7 2.3.1 OTD Library (Java CAPS 6 Repository)
2. JMS JCA to trigger a MDB with a HL7 Delimited message
3. JMS JCA to write result message out
4. JMA MDB to do the processing
5. OTDImporter to provide HL7 2.3.1 OTD and cyustom XSD-based OTD to the EJB for "convenient" mapping

Brief steps to implement this solution are given below. The code will work in Java CAPS 6 Update 1.

Install HL7 2.x OTD Library for the messages that you will deal with.

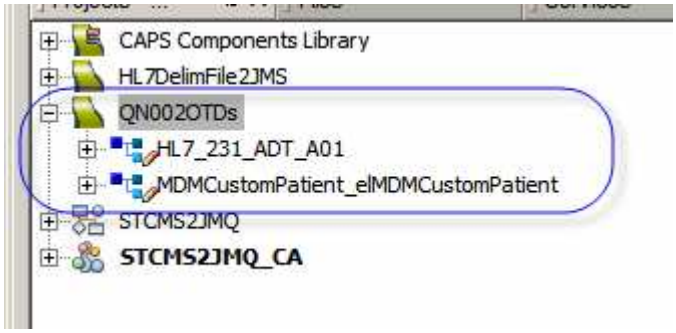
Create a new Classic project

Create a new OTD -> XSD-based OTD from a XML Schema that describes your target XML message. I used the one below.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://xml.netbeans.org/schema/MDMCustomPatient"
            xmlns:tns="http://xml.netbeans.org/schema/MDMCustomPatient"
            elementFormDefault="qualified">
    <xsd:element name="elMDMCustomPatient">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="MSH">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="MSH_3_SENDING_APPLICATION"
type="xsd:string" minOccurs="0" maxOccurs="1" />
                            <xsd:element name="MSH_4_SENDING_FACILITY"
type="xsd:string" minOccurs="0" maxOccurs="1" />
                            <xsd:element name="MSH_7_DATE_TIM_OF_MESSAGE_ISO8601"
type="xsd:string" minOccurs="0" maxOccurs="1" />
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="EVN">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="EVN_1_TRIGGER_EVENT" type="xsd:string"
minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="EVN_5_1_OPERATOR_ID" type="xsd:string"
minOccurs="0" maxOccurs="1"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="PID">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="PID_3X_1_ID" type="xsd:string"
minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="PID_3X_4_ASSIGNING_AUTHORITY"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="PID_3X_5_ID_TYPE_CODE"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="PID_3X_6_ASSIGNING_FACILITY"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="PID_5_1_PATIENT_NAME_FAMILY"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="PID_5_2_PATIENT_NAME_GIVEN"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="PID_5_3_PATIENT_NAME_MIDDLE"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="PID_5_4_PATIENT_NAME_SUFFIX"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="PID_5_5_PATIENT_NAME_PREFIX"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="PID_5_6_PATIENT_NAME_DEGREE"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="PID_7_DATE_TIME_OF_BIRTH_ISO8601"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="PID_8_ADMINISTRATIVE_SEX"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="PID_11X_1_PATIENT_ADDRESS_STREET"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                            <xsd:element
name="PID_11X_2_PATIENT_ADDRESS_OTHER_DESIGNATION" type="xsd:string" minOccurs="0"
maxOccurs="1"/>
                            <xsd:element name="PID_11X_3_PATIENT_ADDRESS_CITY"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="PID_11X_4_PATIENT_ADDRESS_STATE"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                            <xsd:element name="PID_11X_5_PATIENT_ADDRESS_POST_CODE"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
```

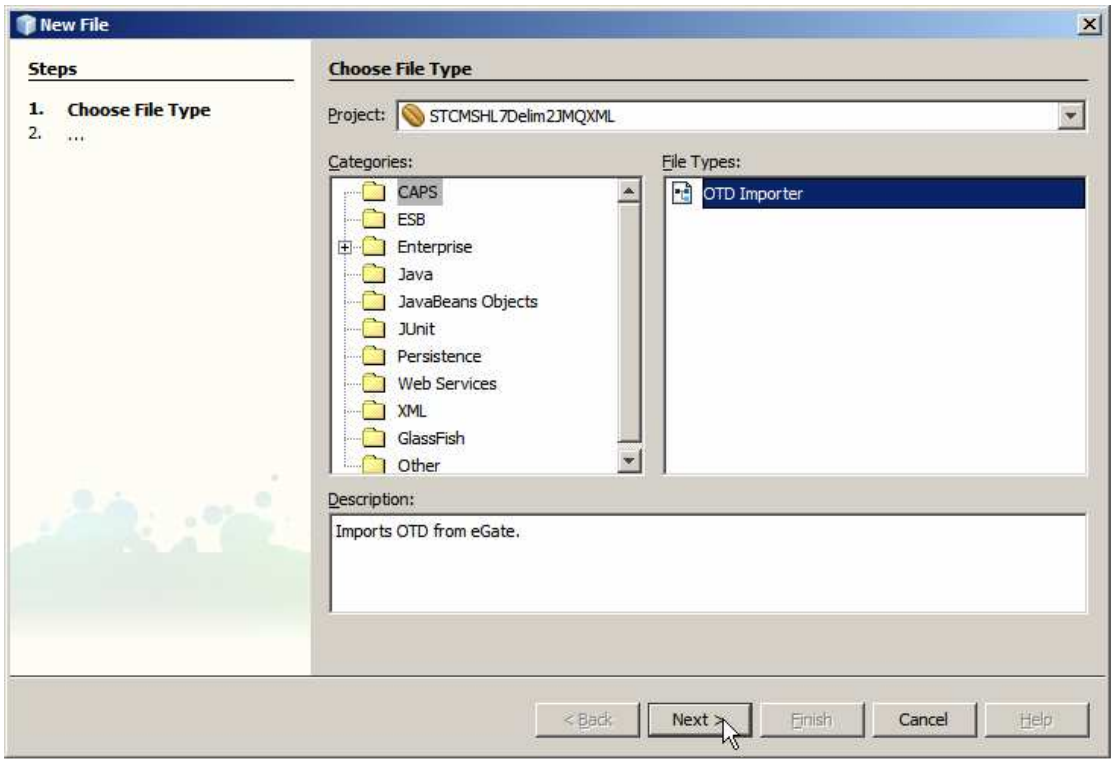
```
                            <xsd:element name="PID_19_MEDICARE_NUMBER"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

From CAPS Components Library -> Message Library -> HL7 -> 2.3.1 (or whatever version you are after) copy the message structure(s) that you need (I used ADT A01) and paste to the OTD project. Strictly speaking this is not necessary but I like to get a private copy.

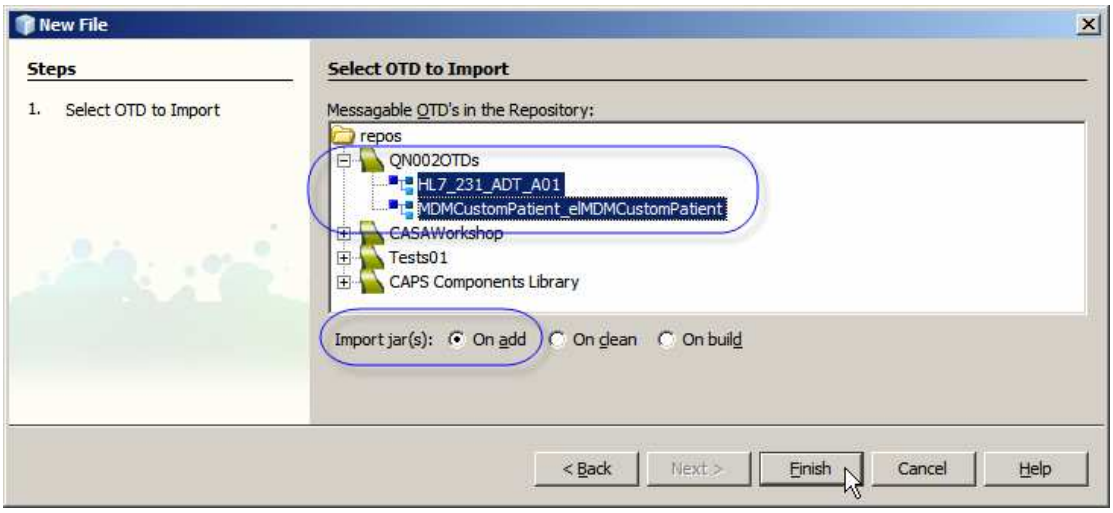


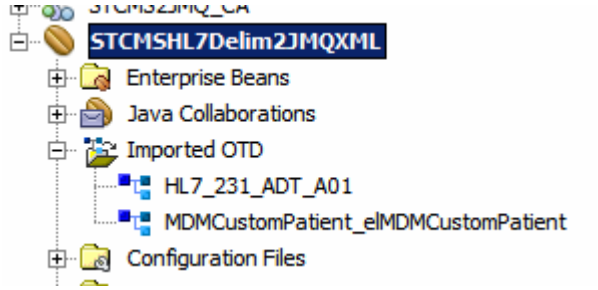
Create a new Enterprise -> EJB Module project, STCMSHL7Delim2JMQXML.

Trigger the New -> Other -> CAPS -> OTD Importer wizard. It will take a little while for list of Repository projects to appear.

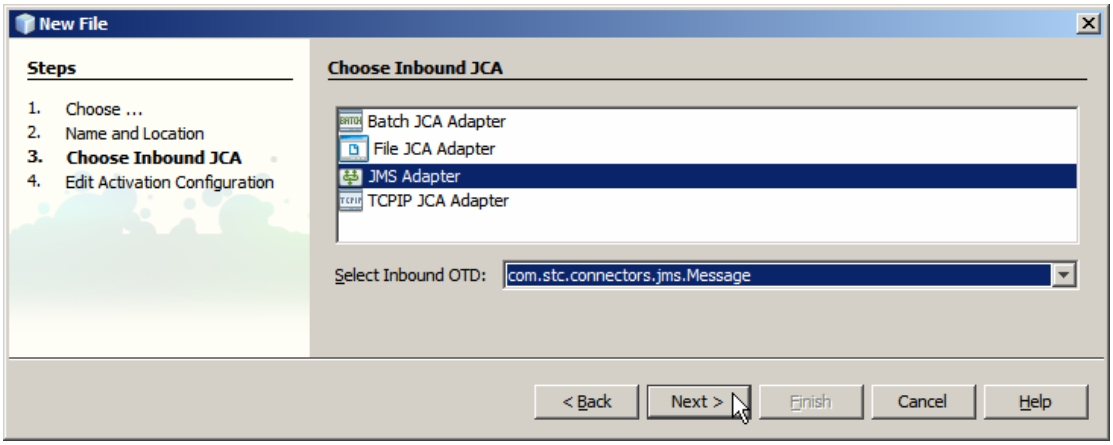


Locate and select the OTDs and make sure to have “On Add” radio button selected. This way no references to the Repository side will be made once the import completes.

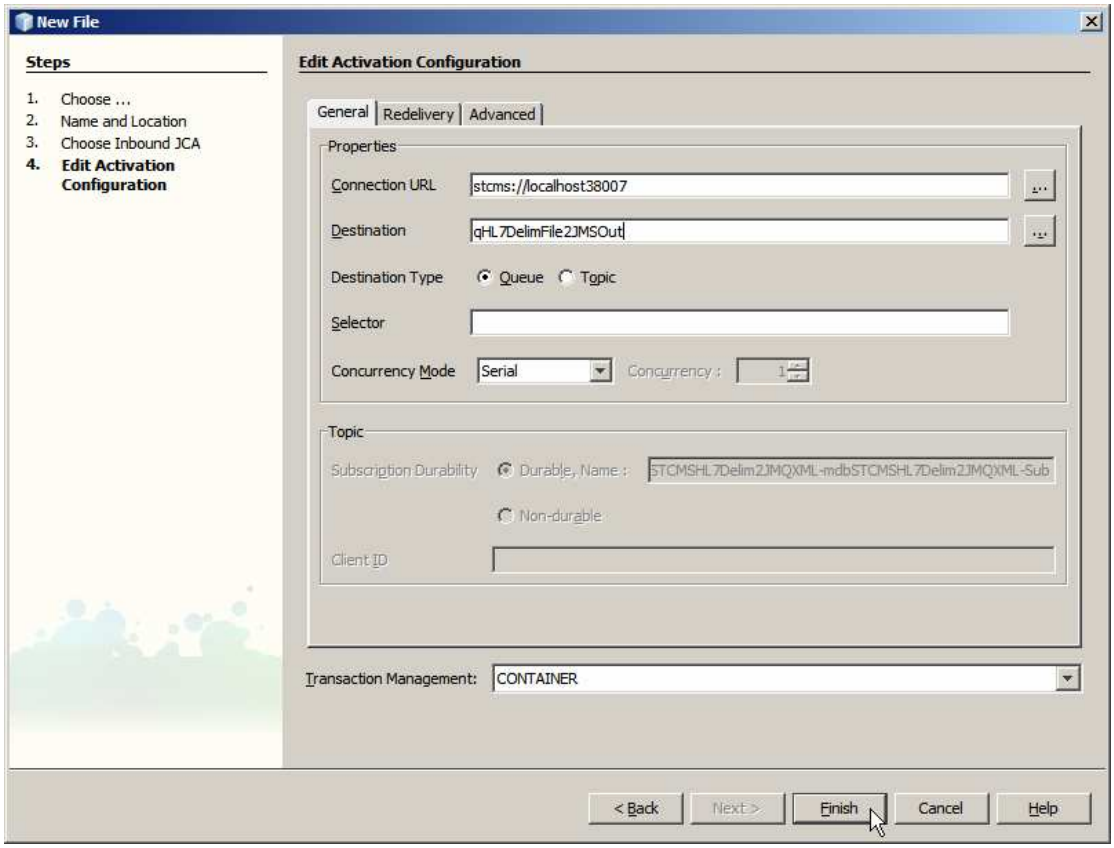




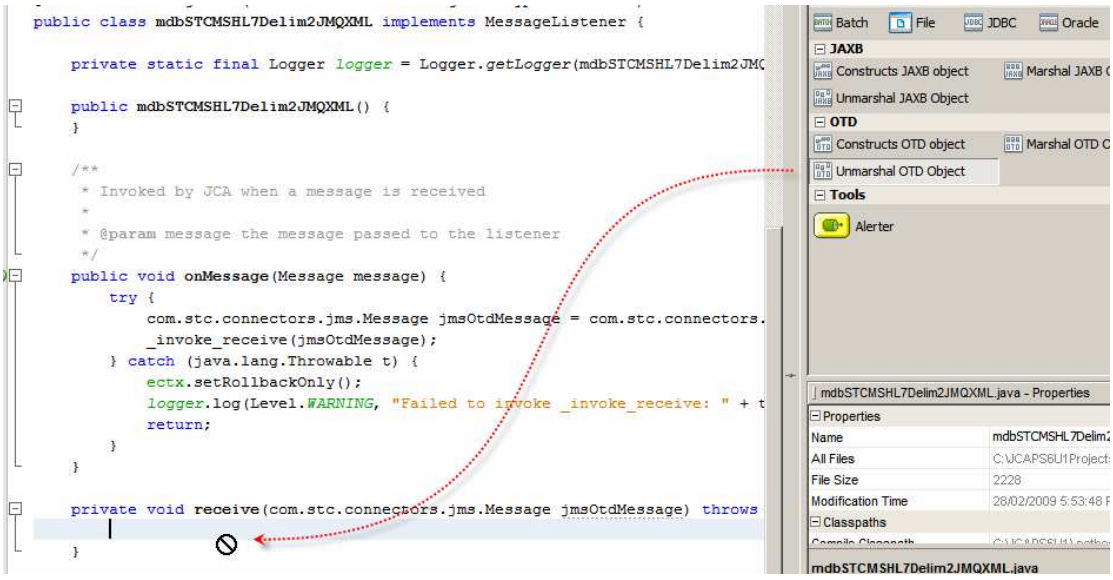
Create a new JCA MDB, mdbSTCMSHL7Delim2JMQXML – pkg.mdbSTCMSHL7Delim2JMQXML. Pick a JMS Adapter as trigger. Make sure to pick com.stc.connectors.jms.Message for inbound oTD.



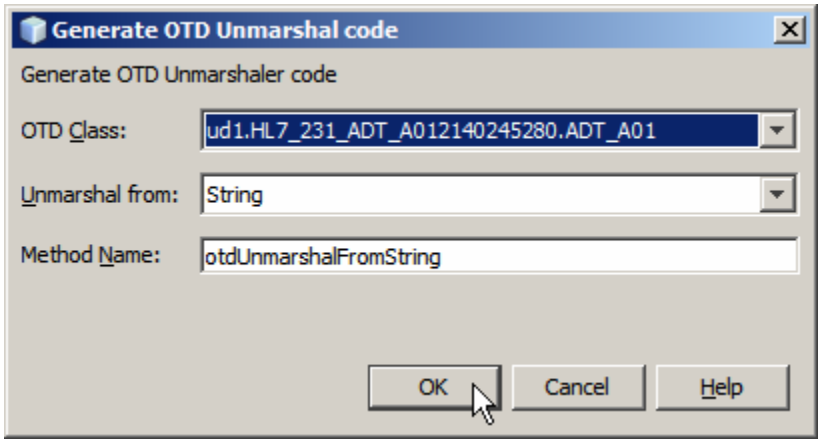
Use stcms://localhost:38007 (or whatever the host name is and whatever the port number if where your STCMS JMS Server is listening). The queue name I picked is the same to which my HL7 delimited messages feeder is sending HL7 delimited messages - qHL7DelimFile2JMSOut.



Drag the “Unmarshal OTD Object” from the palette to the receive method.

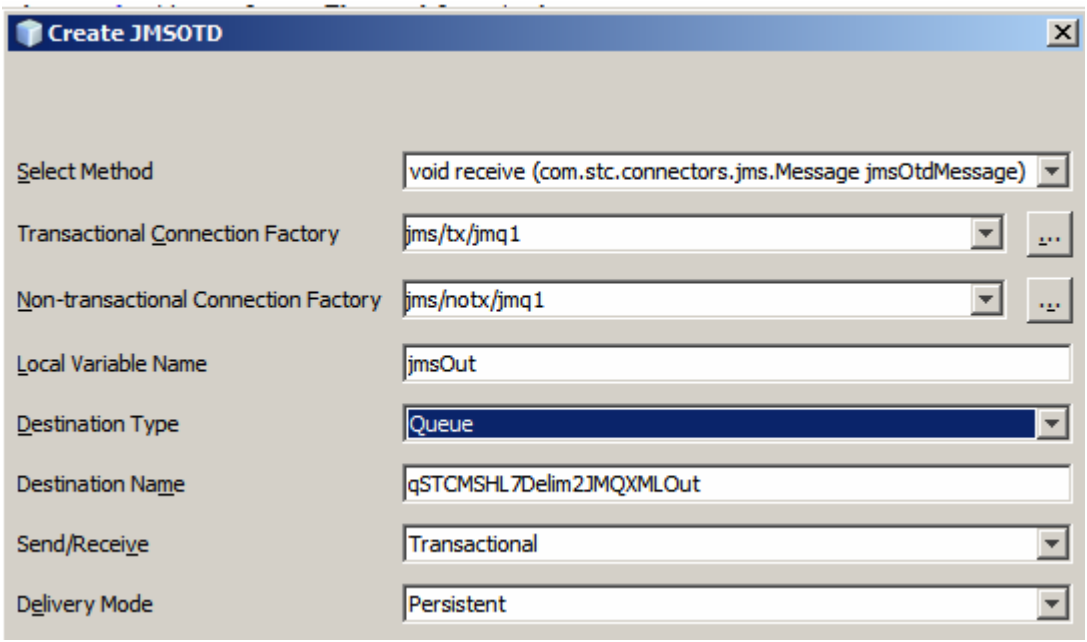


Pick the HL7 OTD.



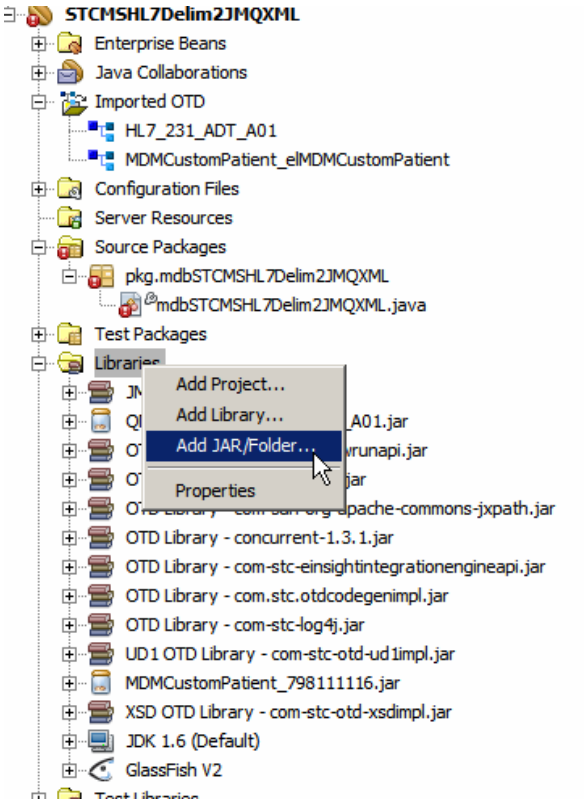
Drag the Marshal OTD Object to the receive method and pick the XSD OTD.

Drag the JMSOTD to the receive method and configure.

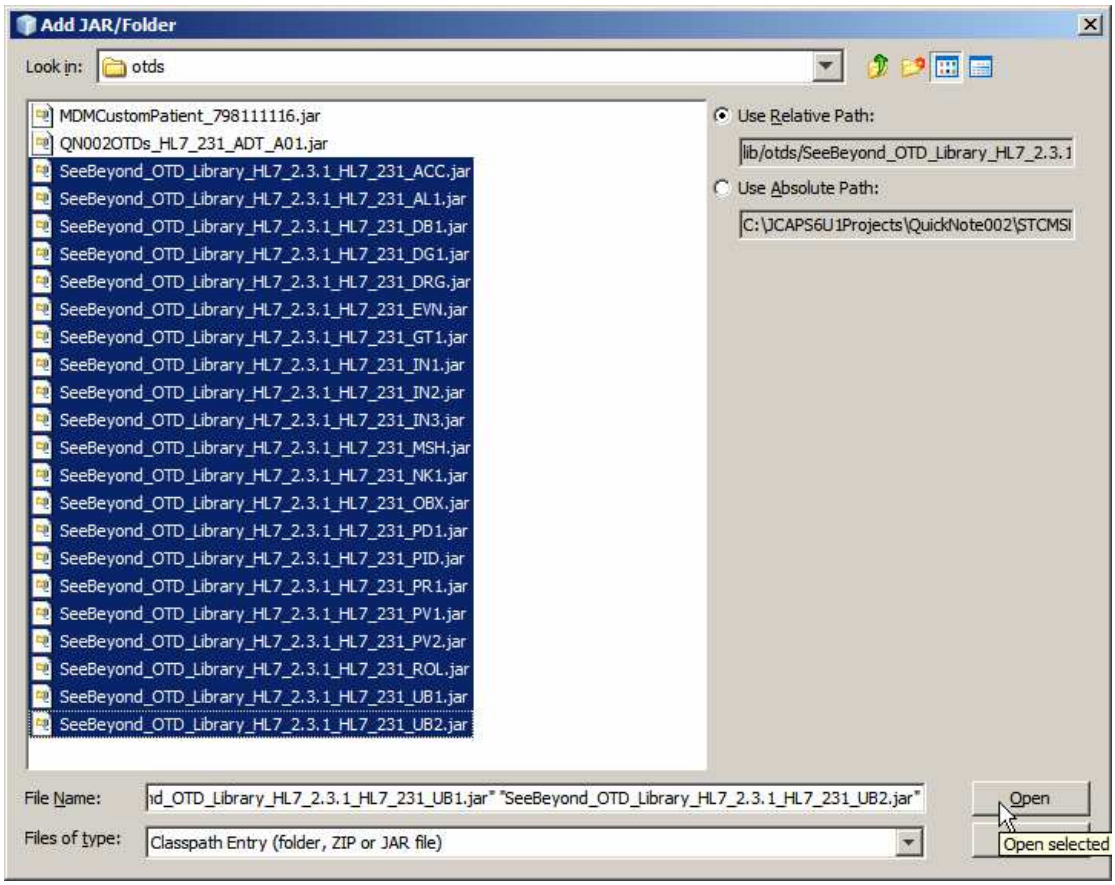


Make sure to pick jms/tx/jmq1 and jms/notx/jmq1 for connection factories to use the Java MQ rather then the stcms. I used qSTCMSHL7Delim2JMQXMLOut as the queue name.

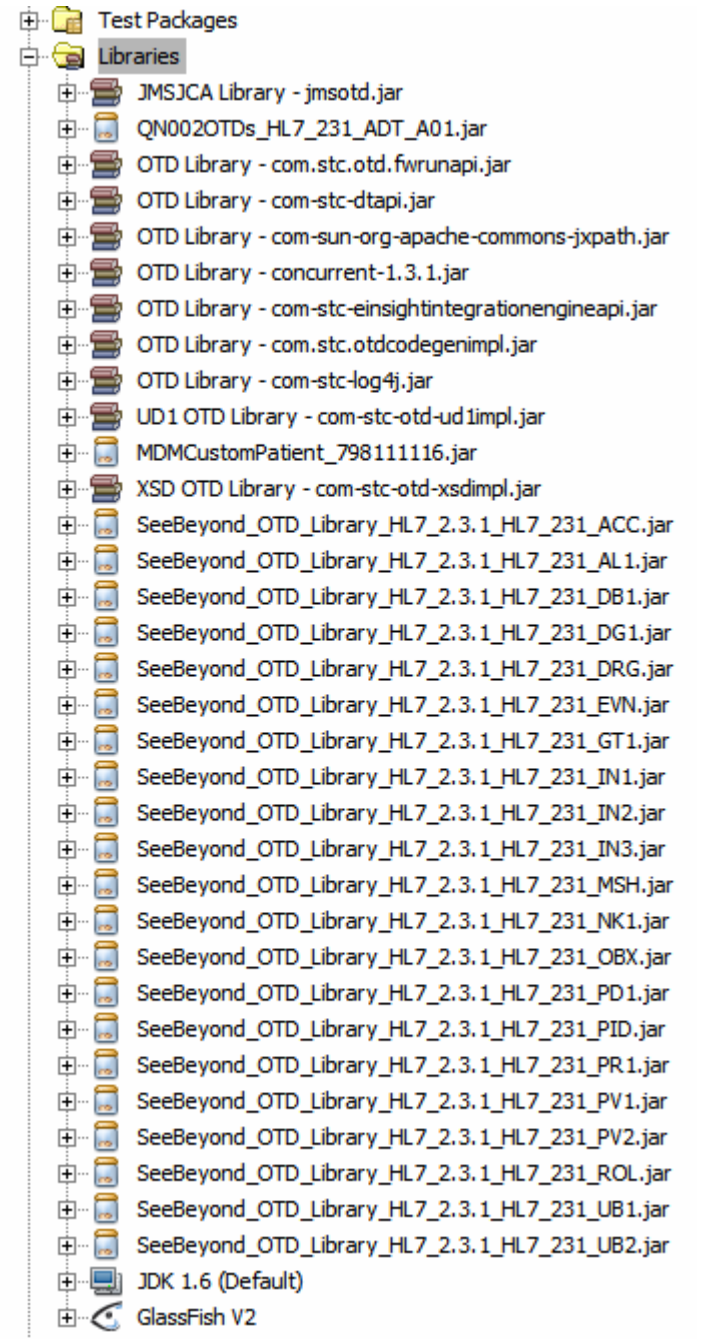
Add all the SeeBeyond JARs, generated but not added to the library.



They will be located in C:\JCAPS6U1Projects\QuickNote002\STCMSHL7Delim2JMQXML\lib\otds or similar for your project and directory hierarchy



It will take a little while for all the libraries to be added.



Add the following import statements:

```
import com.stc.SeeBeyond.OTD_Library.HL7.X_2_3_1.HL7_231_MSH.*;
import com.stc.SeeBeyond.OTD_Library.HL7.X_2_3_1.HL7_231_EVN.*;
import com.stc.SeeBeyond.OTD_Library.HL7.X_2_3_1.HL7_231_PID.*;
import com.stc.SeeBeyond.OTD_Library.HL7.X_2_3_1.HL7_231_PID.Env1.*;
```

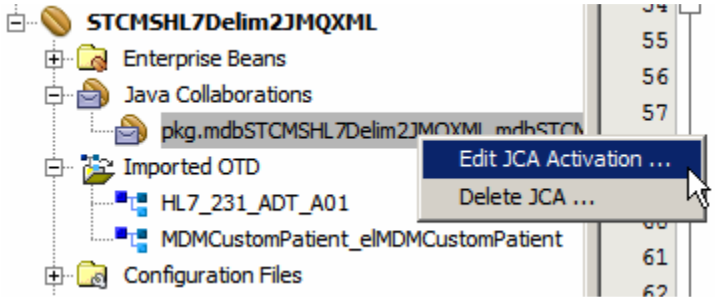
If you are going to use other segments then add the corresponding import statements for them as well. Note that “inner” components are defined in “inner” classes, typically under Evn1. See the imports for the PID segment and its components.

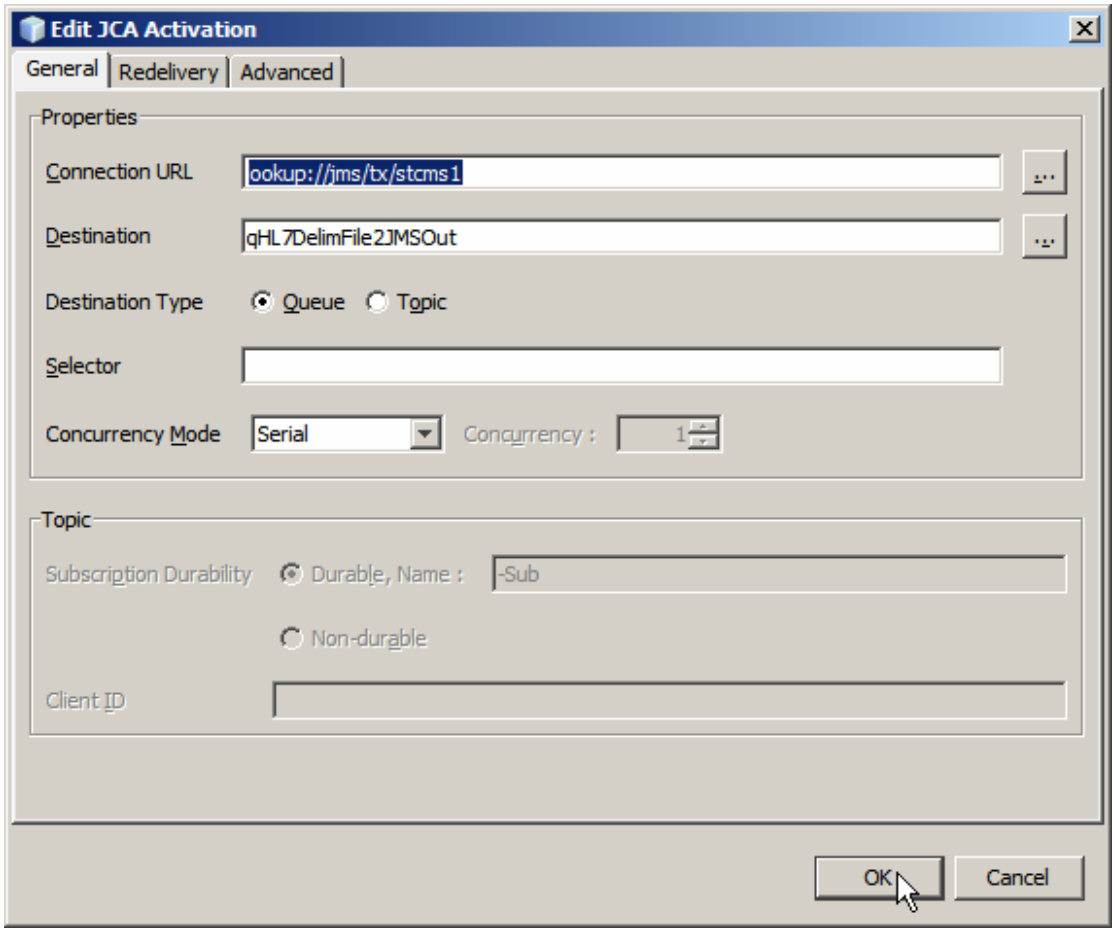
Laboriously add whatever mapping you need, see source below for what I did.

```
    private void receive
                      (com.stc.connectors.jms.Message jmsOtdMessage)
                              throws java.lang.Exception {
        com.stc.connectors.jms.JMS jmsOut
                = com.stc.connectors.jms.JMS.createInstance
                    (_jms_connfact
                    , _notx_jms_connfact
                    , com.stc.connectors.jms.JMS.DestinationType.Queue
                    , "qSTCMSHL7Delim2JMQXMLOut"
                    , true
                    , javax.jms.DeliveryMode.PERSISTENT
                    , 0);

private void receive
        (com.stc.connectors.jms.Message jmsOtdMessage)
            throws java.lang.Exception {
    com.stc.connectors.jms.JMS jmsOut
        = com.stc.connectors.jms.JMS.createInstance
            (_jms_connfact
            , _notx_jms_connfact
            , com.stc.connectors.jms.JMS.DestinationType.Queue
            , "qSTCMSHL7Delim2JMQXMLOut"
            , true
            , javax.jms.DeliveryMode.PERSISTENT
            , 0);

    ud1.HL7_231_ADT_A012140245280.ADT_A01 hl7In
        = new ud1.HL7_231_ADT_A012140245280.ADT_A01();
    hl7In = otdUnmarshalFromString(jmsOtdMessage.getTextMessage());

    xsd.MDMCustomPatient_1832371427.ElMDMCustomPatient xmlOut
        = new xsd.MDMCustomPatient_1832371427.ElMDMCustomPatient();

    xmlOut.getMSH().setMSH_3_SENDING_APPLICATION
        (hl7In.getMSH().getMsh3SendingApplication().getHD().getN341UniversalId());
    xmlOut.getMSH().setMSH_4_SENDING_FACILITY
        (hl7In.getMSH().getMsh4SendingFacility().getHD().getN341UniversalId());

    xmlOut.getEVN().setEVN_1_TRIGGER_EVENT
        (hl7In.getEVN().getEvn1EventTypeCode());
    xmlOut.getEVN().setEVN_5_1_OPERATOR_ID
        (hl7In.getEVN().getEvn5OperatorId(0).getXCN().getN292IdNumberSt());

    xmlOut.getPID().setPID_3X_1_ID
        (hl7In.getPID().getPid3PatientIdentifierList(0).getCX().getN297Id());
//        xmlOut.getPID().setPID_3X_4_ASSIGNING_AUTHORITY
//        (hl7In.getPID().getPid3PatientIdentifierList(0).getCX().getN281AssigningAuth
ority().getHD().getN341UniversalId());
//        xmlOut.getPID().setPID_3X_5_ID_TYPE_CODE
//        (hl7In.getPID().getPid3PatientIdentifierList(0).getCX().getN252IdentifierTyp
eCode());
//        xmlOut.getPID().setPID_3X_6_ASSIGNING_FACILITY
//        (hl7In.getPID().getPid3PatientIdentifierList(0).getCX().getN237AssigningFaci
lity().getHD().getN341UniversalId());

    xmlOut.getPID().setPID_5_1_PATIENT_NAME_FAMILY

(hl7In.getPID().getPid5PatientName(0).getXPN().getN368FamilyLastName().getFN().getN201
FamilyName());
    xmlOut.getPID().setPID_5_2_PATIENT_NAME_GIVEN
        (hl7In.getPID().getPid5PatientName(0).getXPN().getN22GivenName());
//        xmlOut.getPID().setPID_5_3_PATIENT_NAME_MIDDLE
//     (hl7In.getPID().getPid5PatientName(0).getXPN().getN368FamilyLastName().getFN().
getN201FamilyName());
//        xmlOut.getPID().setPID_5_4_PATIENT_NAME_SUFFIX
//          (hl7In.getPID().getPid5PatientName(0).getXPN().getN273SuffixEGJrOrIii());
//        xmlOut.getPID().setPID_5_5_PATIENT_NAME_PREFIX
//          (hl7In.getPID().getPid5PatientName(0).getXPN().getN235PrefixEGDr());
//        xmlOut.getPID().setPID_5_6_PATIENT_NAME_DEGREE
//          (hl7In.getPID().getPid5PatientName(0).getXPN().getN203DegreeEGMd());
```

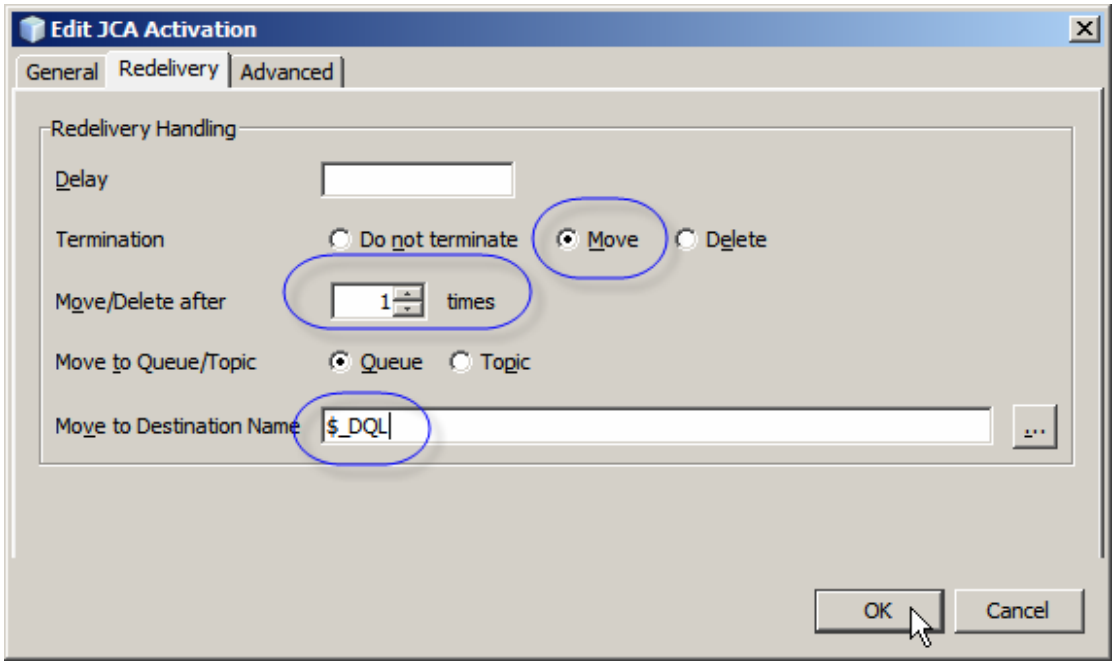
```
//        xmlOut.getPID().setPID_7_DATE_TIME_OF_BIRTH_ISO8601
//          (hl7In.getPID().getPid7DateTimeOfBirth().getTS().getN439TimeOfAnEvent());
    xmlOut.getPID().setPID_8_ADMINISTRATIVE_SEX
        (hl7In.getPID().getPid8Sex());
    // ... and so on and so on ...

    jmsOut.sendText(otdMarshalToString(xmlOut));

}
```

If you followed the steps so far, when you build and deploy this project you will get an exception because I did not specify credentials for the STCMS URL. Instead of providing stcms://host:port provide the jndi reference, lookup://jms/tx/stcms1.





When at it, configure redelivery handling so that an issue in processing the message will not cause infinite redelivery loop.



Now build and deploy the project.

Create a repository-based feeder project which reads a HL7 message from a file and sends it to a JMS Queue, qHL7DelimFile2JMSOut, for the JCA project to pick up.
The Java CAPS 6 Repository project export is provided as a ZIP archive.
A sample HL7 ADT A01 message is provided as a ZIP archive.
The JCA project is provided as a ZIP archive.

Contrive to make the file containing the ADT A01 available to the feeder project, so that it can send it to JMS, and observe the XML message in the Java MQ Queue, qSTCMSHL7Delim2JMQXMLOut.

If all is well, the Enterprise Manager will show the XML message, derived from the HL7 delimited message, using the project discussed above.