GlassFish ESB v2.1

Intelligent Event Processor in Healthcare

"Excessive Length of Stay" Example

Michael.Czapski@sun.com

September, 2009

Table of Contents

Introduction

As a healthcare enterprise looks after patients, information is gathered about various events that take place. Information about notable events, Admissions and Discharges, for example, is recorded in Hospital Information Systems or Patient Administration Systems. These systems typically broadcast event information in a form of HL7 messages for use by other enterprise systems, for example laboratory or diagnostic imaging. A stream of HL7 messages can be intercepted and processed to derive all sorts of interesting information.

The solution developed in this walkthrough deals with Excessive Length of Stay. Length of stay is defined as the period between patient's admission to and discharge from the hospital. Statistical average expected length of stay is typically available for different kinds of patients presenting with different kinds of conditions. A significant variation from the average length of stay for specific patients may indicate complications, treatment errors, infections and other kinds of issues that the hospital needs to investigate. Notification of such incidents may help the hospital in addressing these issues and prevent future occurrences.

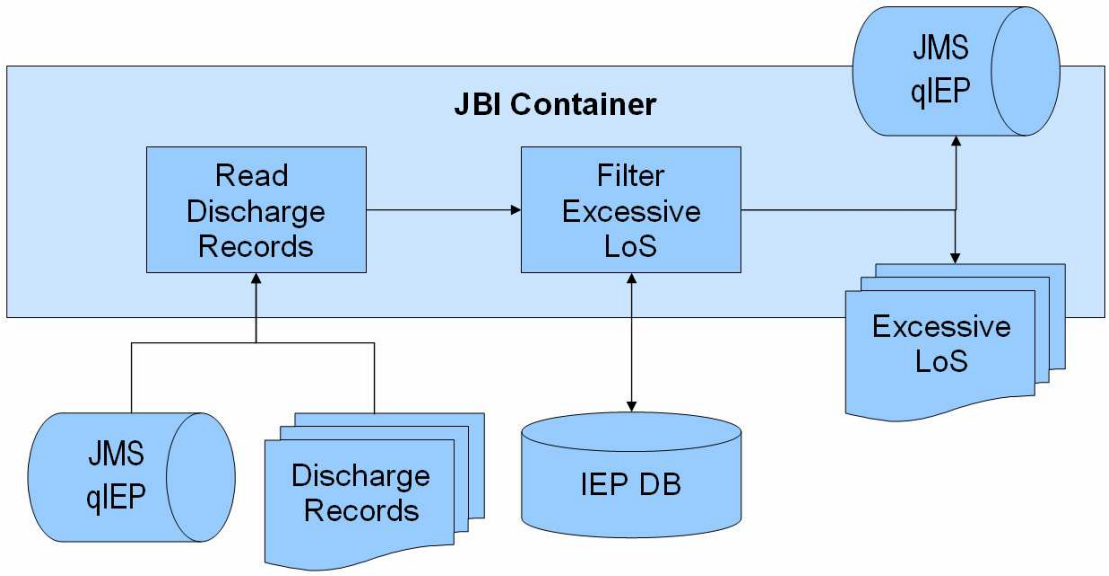
In this solution the Intelligent Event Processor is used to calculate the continuously updated average length of stay over a period of time and use it to compare against each event's length of stay. It passes, to the downstream component, all events where the length of stay exceeds the average by 1 ½ times and ignores all others.

In the initial iteration, the solution reads a stream of discharge messages, containing admission date, discharge date, length of stay, and a bunch of other fields from a file and passes them to the IEP process. The IEP process keeps the window on the last 10 seconds worth of records and continuously calculates the average length of stay over all records in that window. As records are added to and removed from the window the average is recalculated. As each record is seen its length of stay is compared to the average length of stay of all records in the window at the time. If the length of stay in the current record is less then or equal to 1 ½ times the average at the same time the record is discarded. If the average is greater the record is ejected to the output and ultimately written to a file of exception records.

In a subsequent iteration the solution is modified to accept messages from a JMS Queue. This modification allows the solution to use the stream of discharge messages produced by the HL7 Processor solution, discussed in "HL7 Processor Demonstration - GlassFish ESB v2.1", http://blogs.sun.com/javacapsfieldtech/entry/hl7_processor_demonstration_glassfish_esb.

In a further modification the solution is configured to send notification messages to another JMS Queue. Notification messages are processed by a different solution and sent to an email recipient.

The final solution is shown in a schematic below. Neither the HL7 Processor solution, nor the Notification solutions are shown in the diagram. The former is discussed elsewhere. The later is developed in this walkthrough.



Building the Solution

Preliminaries

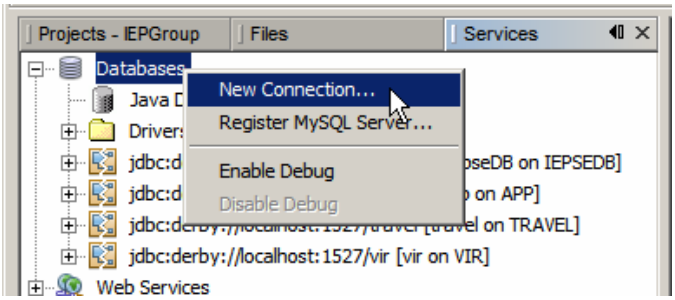
The key to untroubled start with IEP is to make sure the initial startup of components is performed correctly. So, once off, just after installing a new version of GlassFish ESB, do the following:

1. start NetBeans IDE
2. start the Java DB from the Services Tab in the IDE
3. start the GlassFish app server from the services tab in the IDE

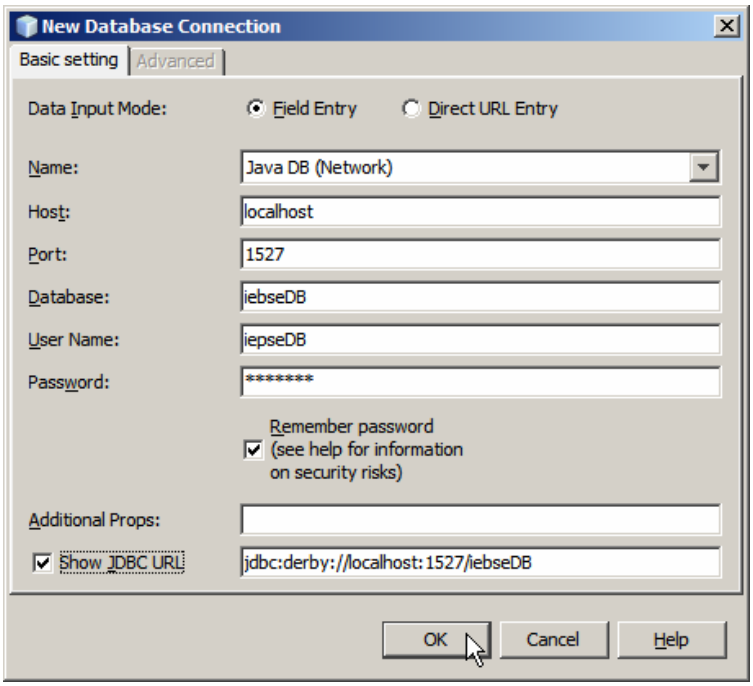
4. start the sun-iep-engine form the JBI Service Engines node tree in the IDE

Once these steps are performed in the correct order an initial iepseDB database is created. Thereafter one can start components form command scripts and suchlike. I typically start the Java DB using a script, then the App Server using a script, then the NetBeans IDE using a script.

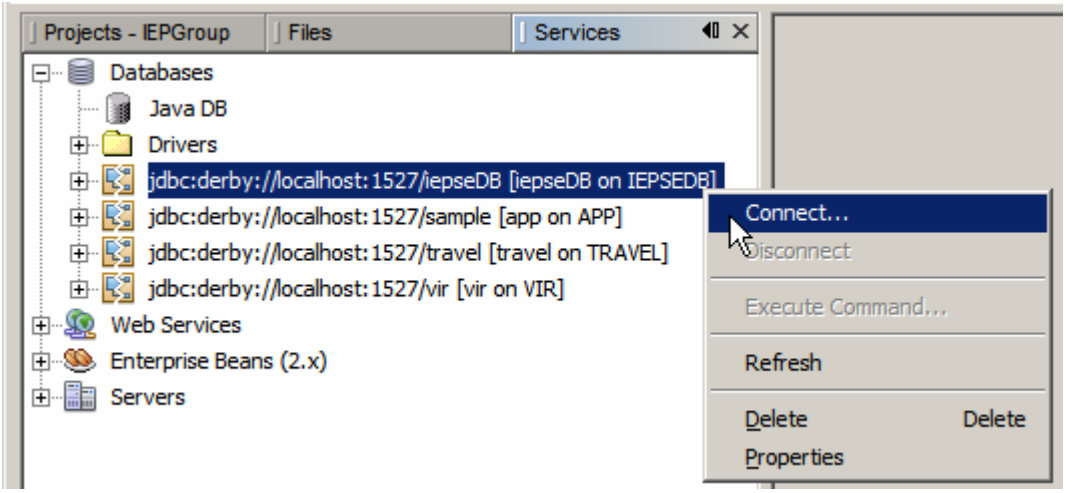
Make sure that the IEP DB is available. In NetBeans IDE switch to the Services Tab, right-click Databases node and choose “New Connection”.



Complete details (localhost, 1527, iepseDB, iepseDB, iepseDB)

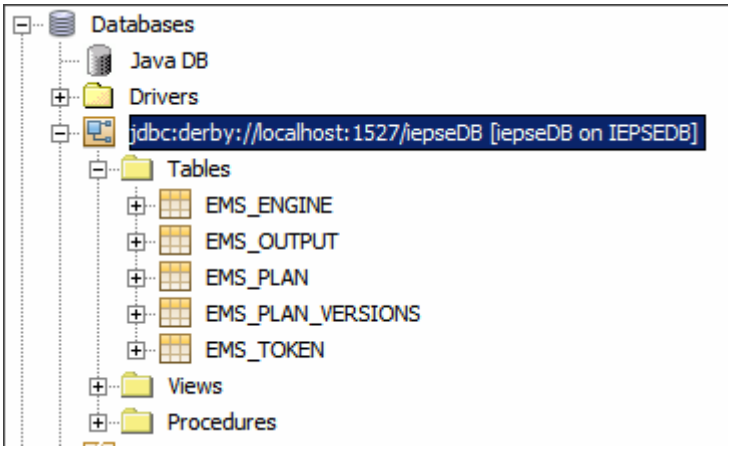


Right-click the new connection and choose Connect …



If configuration is correct, the database server is running and the iepseDB has been created, the connection will succeed. If not, troubleshoot and fix what issues there may be until this activity is successful.

Confirm the presence of IEP basic tables.



Before using the iepseDB make a backup copy of it (the directory containing the iepseDB files). This should be done when the database server is not running. Sometimes iepseDB gets corrupted and needs to be restored.

Prerequisites

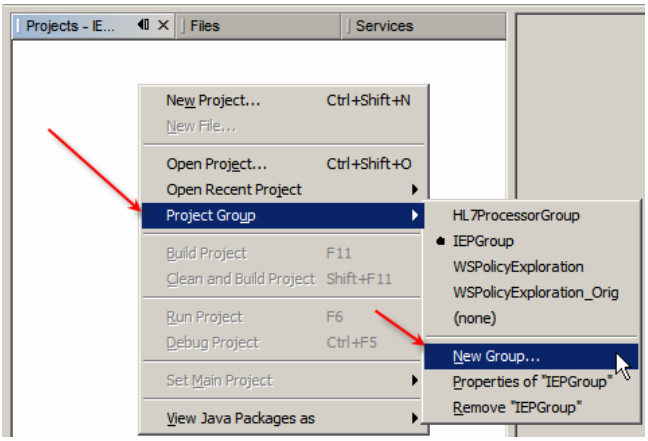
The XML Schema, IEPCustomDischarge.xsd, see directory Combined/prerequisites, is the schema to which records in the input files conform.

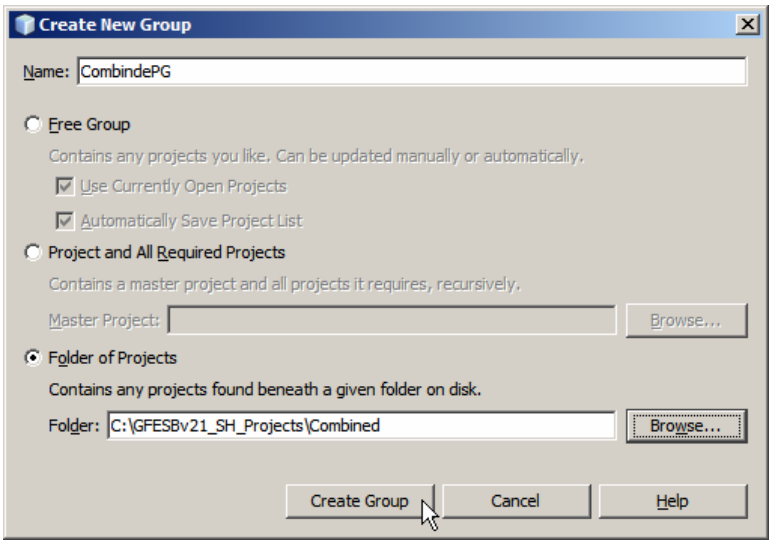
```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.netbeans.org/schema/IEPCustomDischarge"
    xmlns:tns="http://xml.netbeans.org/schema/IEPCustomDischarge"
    elementFormDefault="qualified"
    >
    <xsd:element name="elIEPCustomDischarge">
    <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"
                    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_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_7_DATE_TIME_OF_BIRTH"
                    type="xsd:string" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="PID_8_ADMINISTRATIVE_SEX"
                    type="xsd:string" minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="PV1">
            <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="PV1_19_1_VISIT_NUMBER"
                    type="xsd:string" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="PV1_44_ADMIT_DATE_TIME"
                    type="xsd:string" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="PV1_45_DISCHARGE_DATE_TIME"
                    type="xsd:string" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="LOS"
                    type="xsd:int"/>
            </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Create a Project Group

Right-click on the blank space in the Project Explorer, choose “Project Group” -> “New Group …”, named CombinedPG, based in a folder named Combined. If the project group already exists switch to it instead, to add new projects

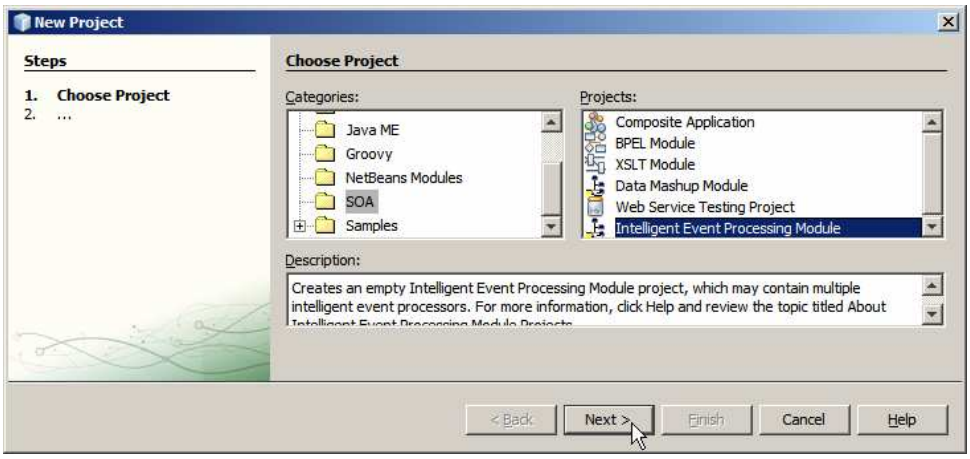




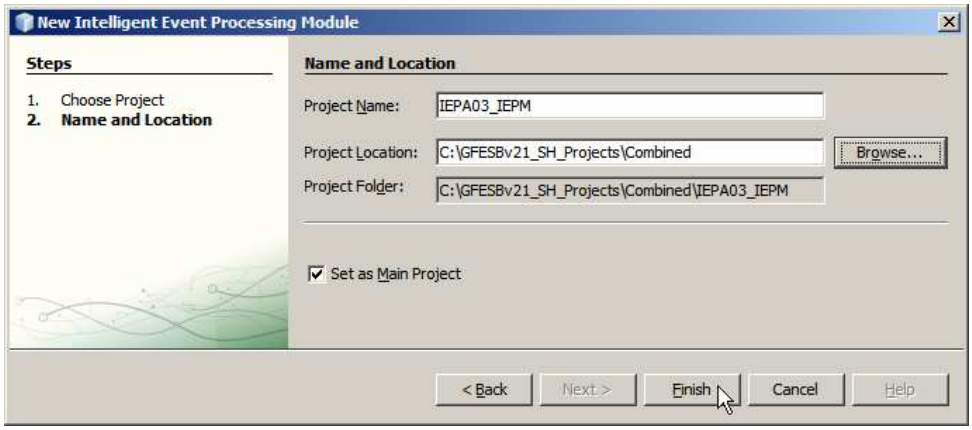
All our projects will be created in this project group and will appear under the specified folder in the file system.

Building IEP Project

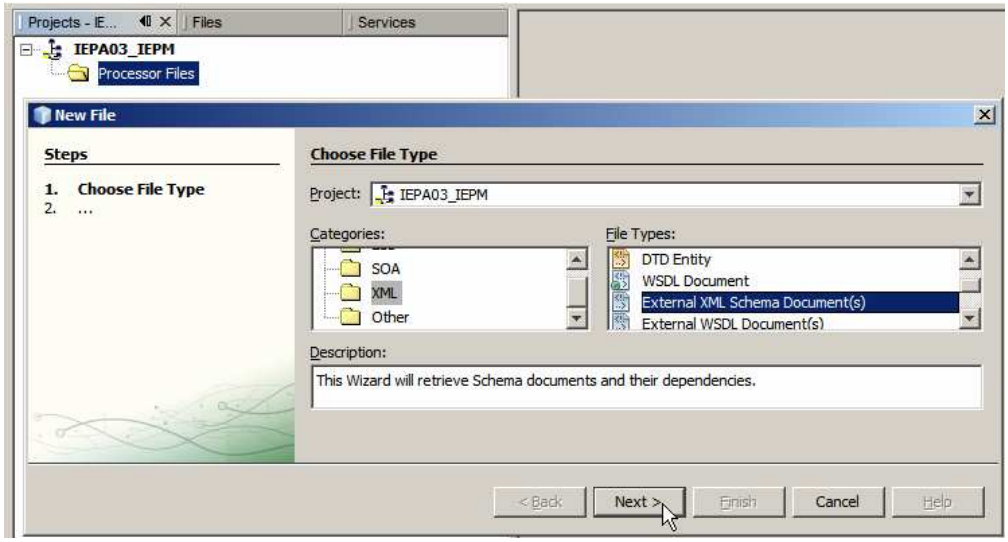
Right-click in the empty space in the Project Explorer and create “New Project” -> “SOA” -> “Intelligent Event Processing Module”.

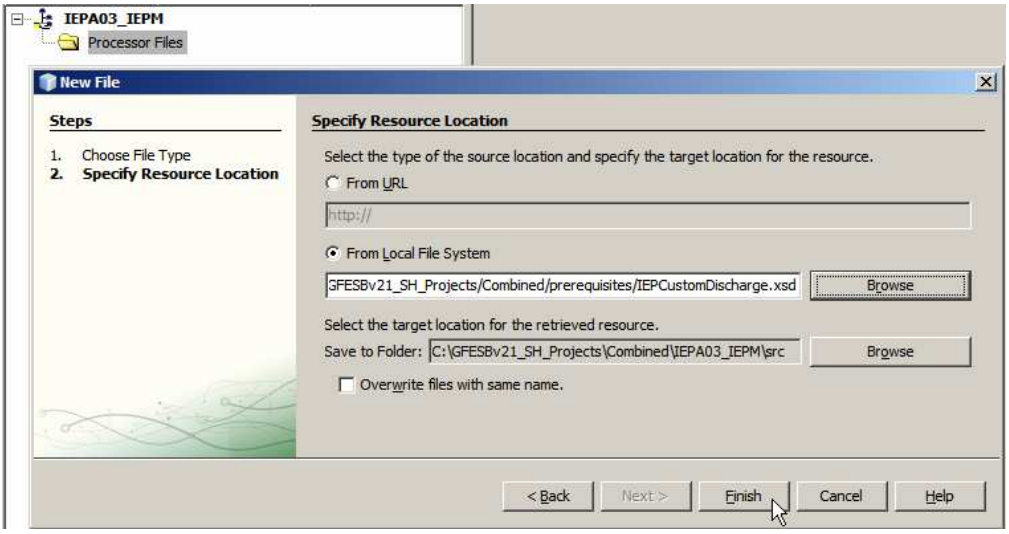


Name the project IEPA03_IEPM.

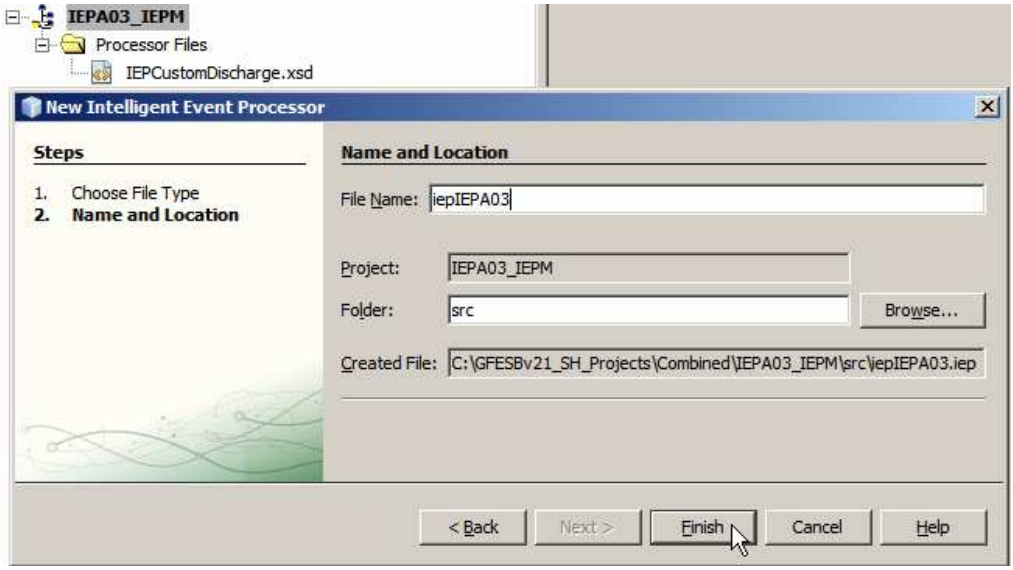


Right-click on the “Processor Files” folder under the new module name, choose “New” -> “Other” -> “XML” -> “External XML Schema Documents(s)”, click the “From File System” radio button, navigate to `Combined/prerequisites/IEPCustomDischarge.xsd` and select it.

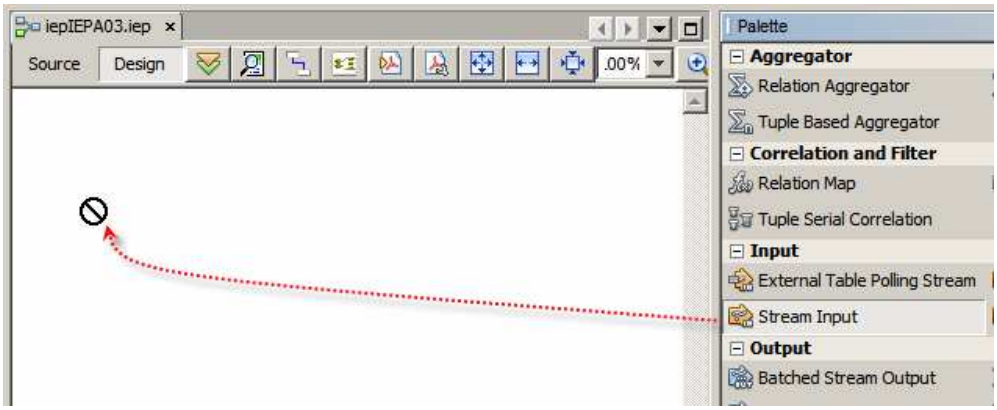




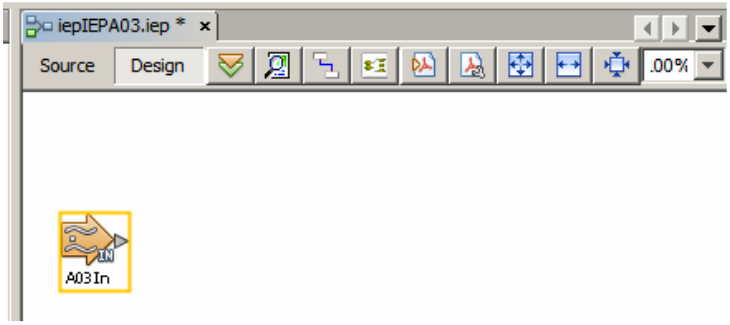
Right-click the name of the project, choose “New” -> “Intelligent Event Processor …”. Name the processor iepIEPA03.



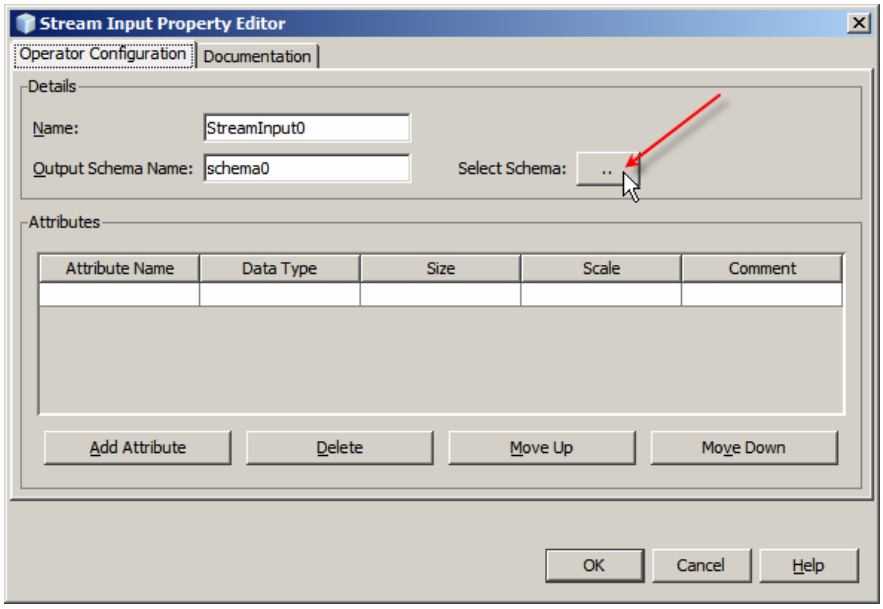
Drag the “Input” -> “Stream Input” operator from the Palette onto the canvas



Rename the operator to A03In.

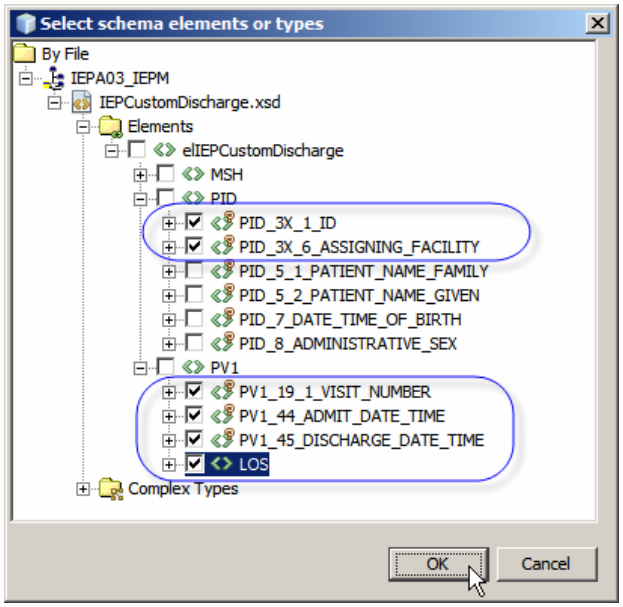


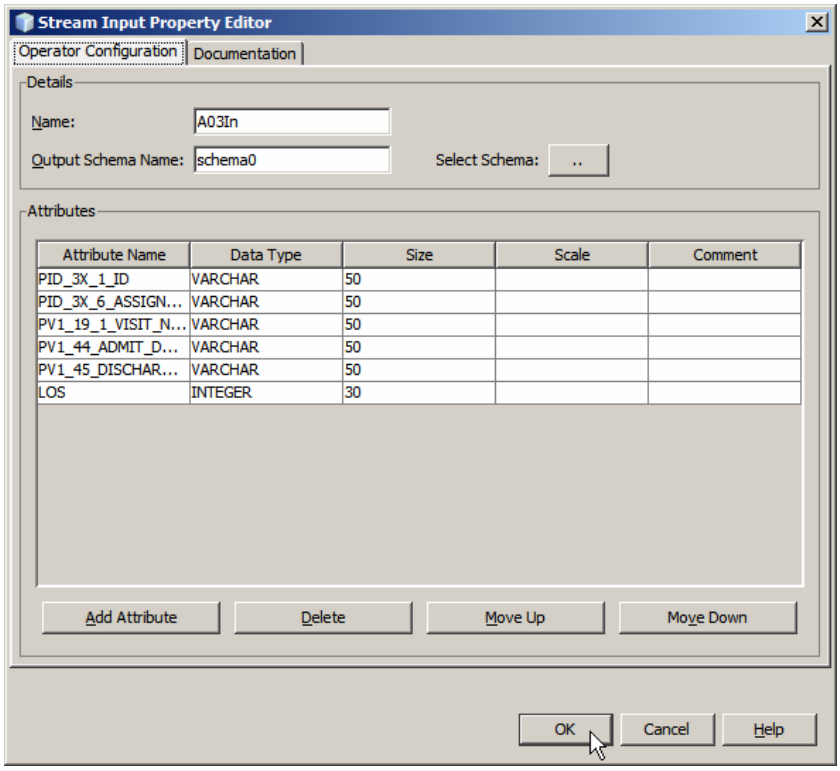
Double-click on the operator on the canvas to open its properties. Click the “select schema” button.



Expand past the root element of the schema and choose

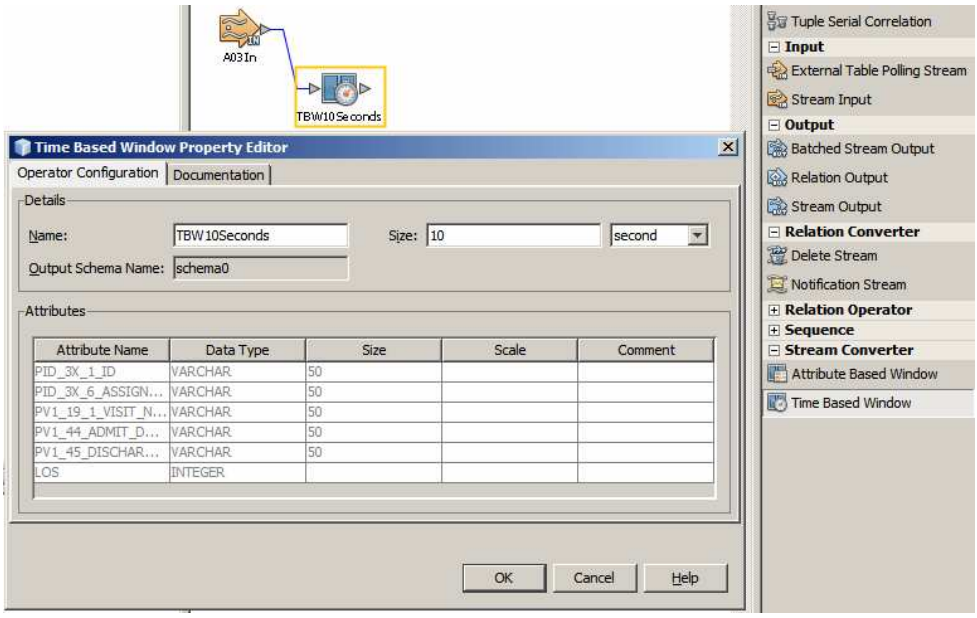
pid_3x_1_id
pid_3x_6_assigning_facility
pv1_19_1_visit_number
pv1_44_admit_date_time
pv1_45_discharge_date_time
los





Close the dialogue box.

Drag “Stream Converter” -> “TimeBasedWindow” operator onto the canvas, rename it to TBW10Seconds, connect A03In to TBW10Seconds, edit properties of TBW10Seconds and configure “Size” to 10 seconds.

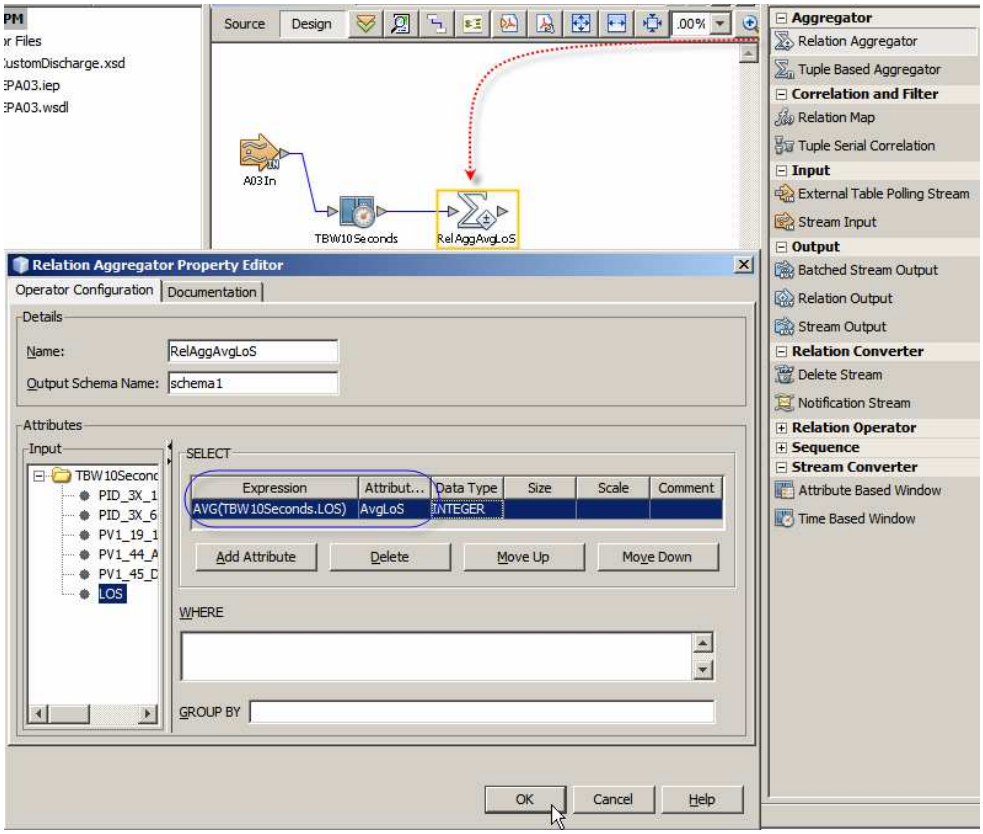


Drag “Aggregator” -> “Relational Aggregator” operator to the canvas, rename it to RelAggAvgLoS and connect to TBW10Seconds.

Double-click RelAggAvgLoS to edit properties.

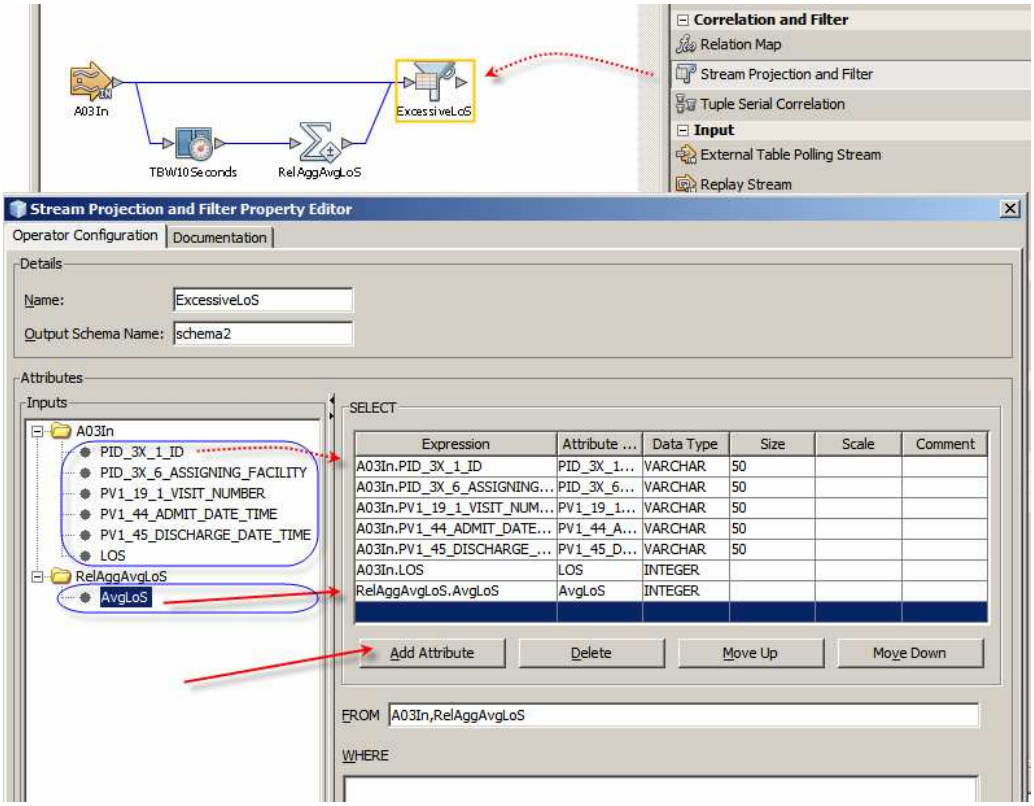
Drag LOS Input Attribute to the SELECT expression field and surround it by AVG() to form an expression: AVG(TBW10Seconds.LOS)

Change attribute name to AvgLoS.

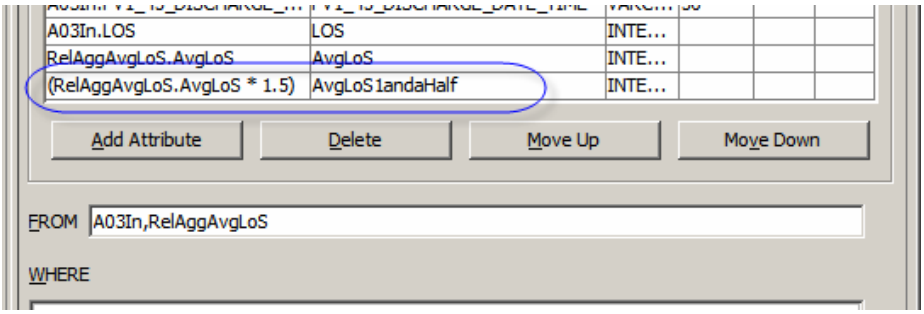


Drag “Correlation and Filter” -> “Stream Projection and Filter” onto the canvas. Rename it to ExcessiveLoS. Connect to RegAggAvgLoS and to A03In

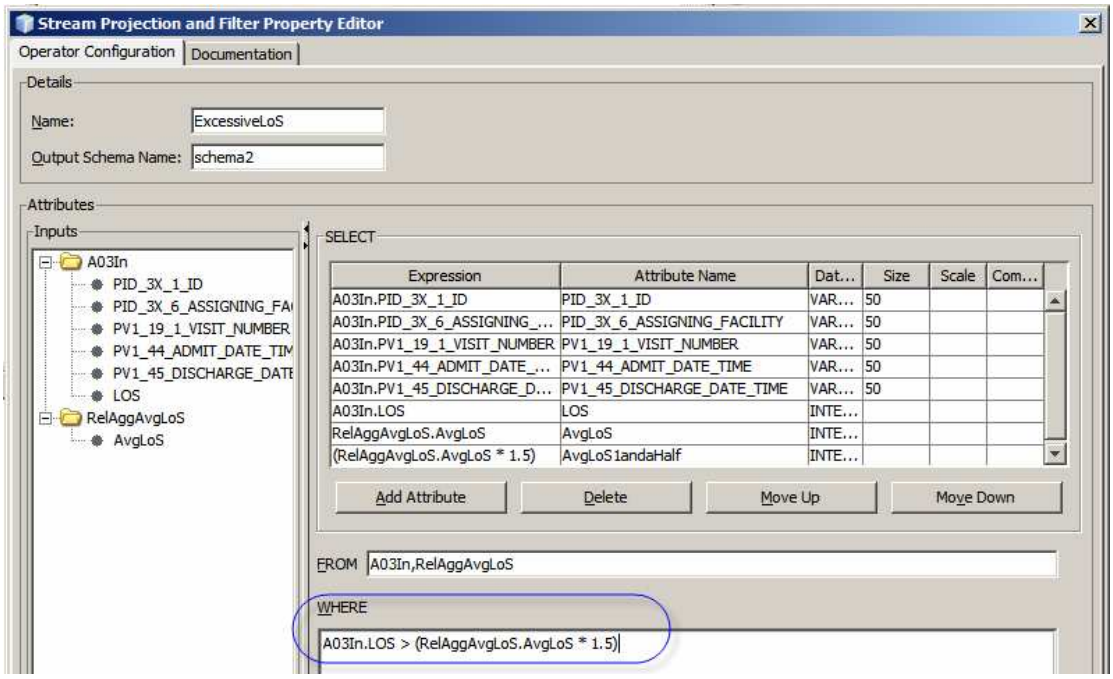
Edit properties of the ExcessiveLoS operator. Add all input attributes to the SELECT table and add one more empty attribute.



Configure the empty line to an expression (RelAggAvgLoS.AvgLoS * 1.5), name the attribute AvgLoS1andaHalf and set its Data Type to INTEGER

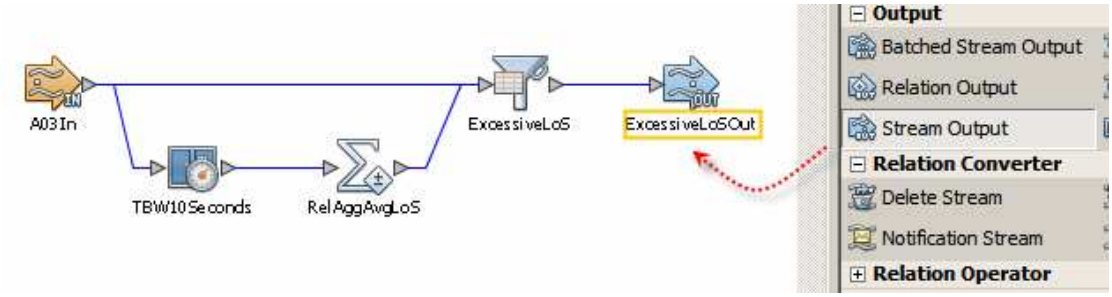


Configure the WHERE to expression by entering A03In.LOS > (RelAggAvgLoS.AvgLoS * 1.5)



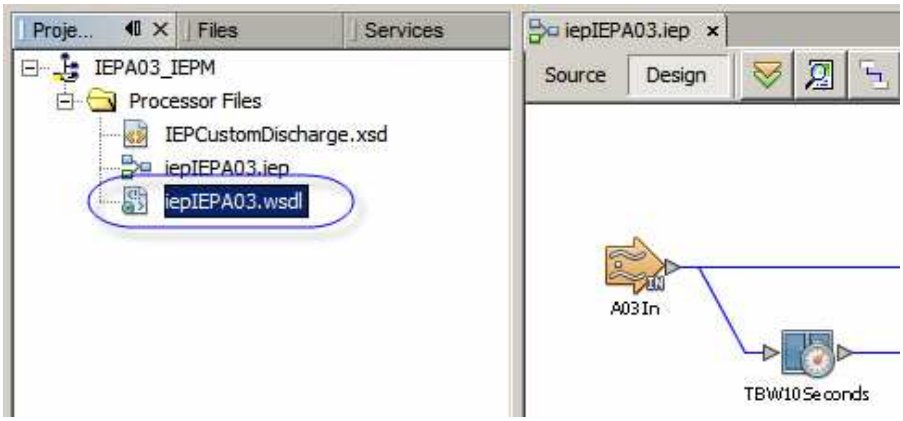
Close the dialogue box.

Drag a “Output” -> “Stream Output” onto the canvas, name the stream ExcessiveLoSOut and connect ExcessiveLoS to ExcessiveLoSOut.

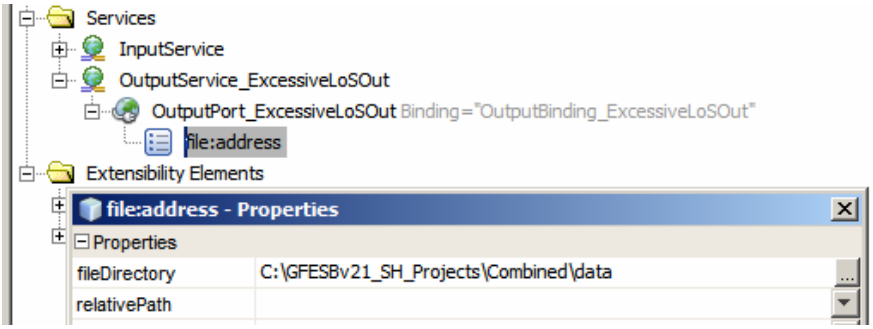


Build the project.

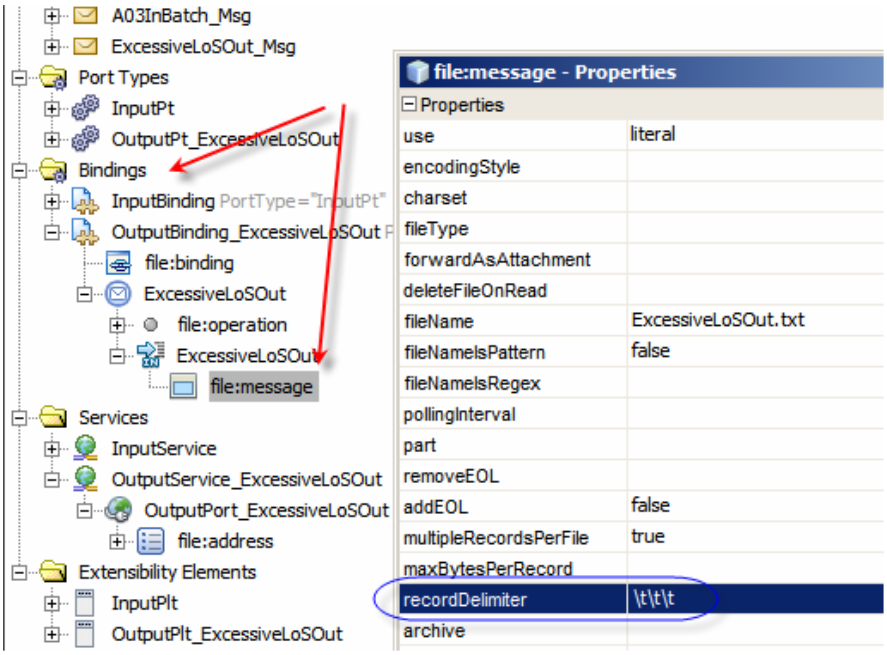
Observe new WSDL artifact in the project explorer.



Edit the generated WSDL and modify OutputService_ExcessiveLoSOut and properties. Change file:address: to <project folder>/Combined/data.



Change file:message: recordDelimiter to “\t\t\t”.



Build the project again.

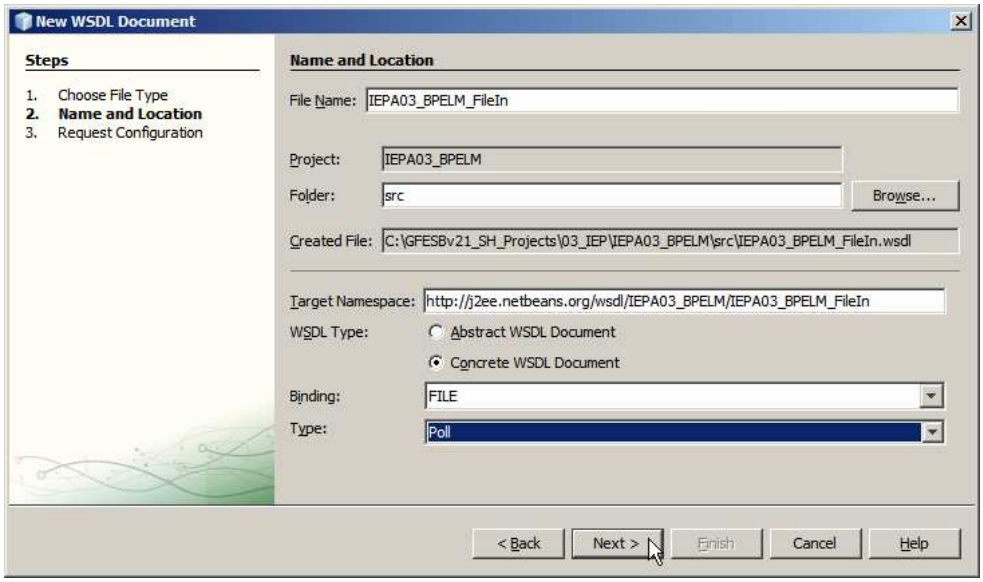
The IEP project, first iteration, is ready

Building BPEL Project

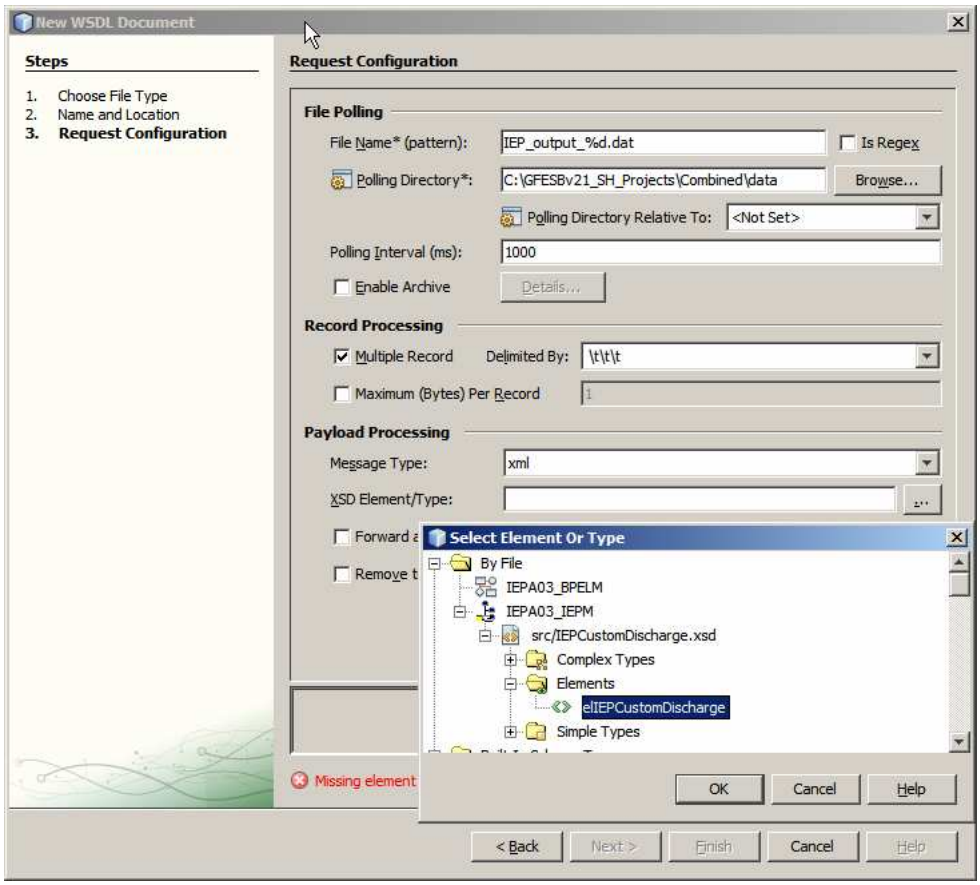
Create a “New Project” -> “SOA” -> “BPEL Module” and name it IEPA03_BPELM.

Right-click the name of the BPEL module project and create “new” -> “WSDL Document”.

1. name IEPA03_BPELM_FileIn
2. WSDL Type: concrete wsdl document
3. binding: file
4. type: poll



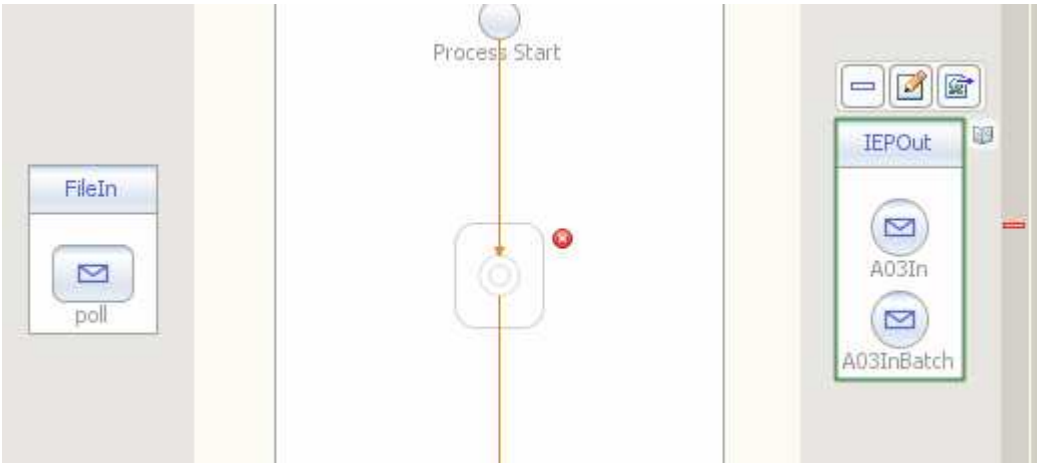
5. set file name pattern to IEP_output_%d.dat
6. set path to <project folder>/Combined/data
7. set multiple record: true
8. set Delimited by: **\t\t\t**
9. set Message Type: XML
10. For XSD Element Type choose XSD from the IEPA03_IEPM project: elIEPCustomDischarge



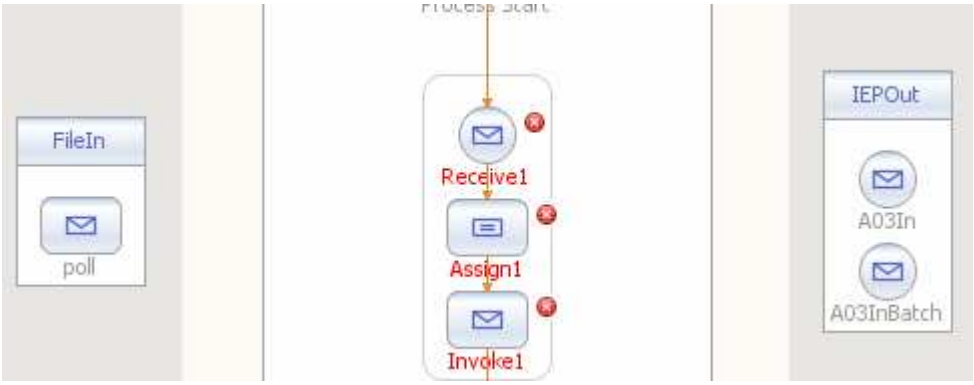
Open IEPA03_BPELM.bpel process model.

Drag IEPA03_BPELM_FileIn WSDL onto the left-hand swim line and name the partner link FileIn.

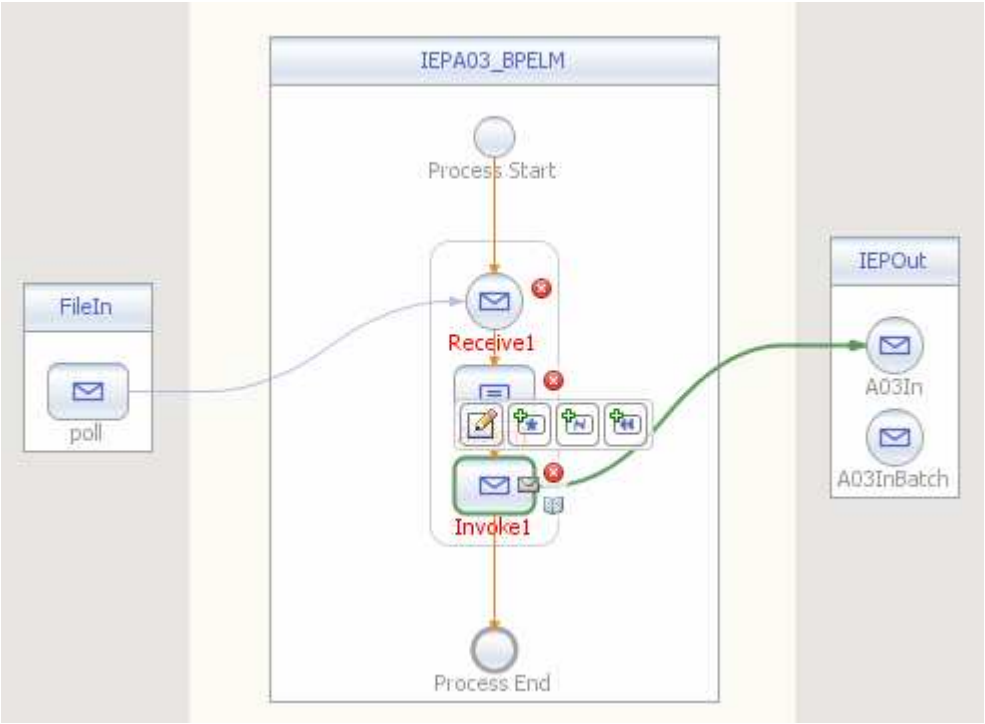
Drag iepIEPA03.wsdl from the IEP03_IEPM module onto the right-hand swim line and name the partner link IEPOut.



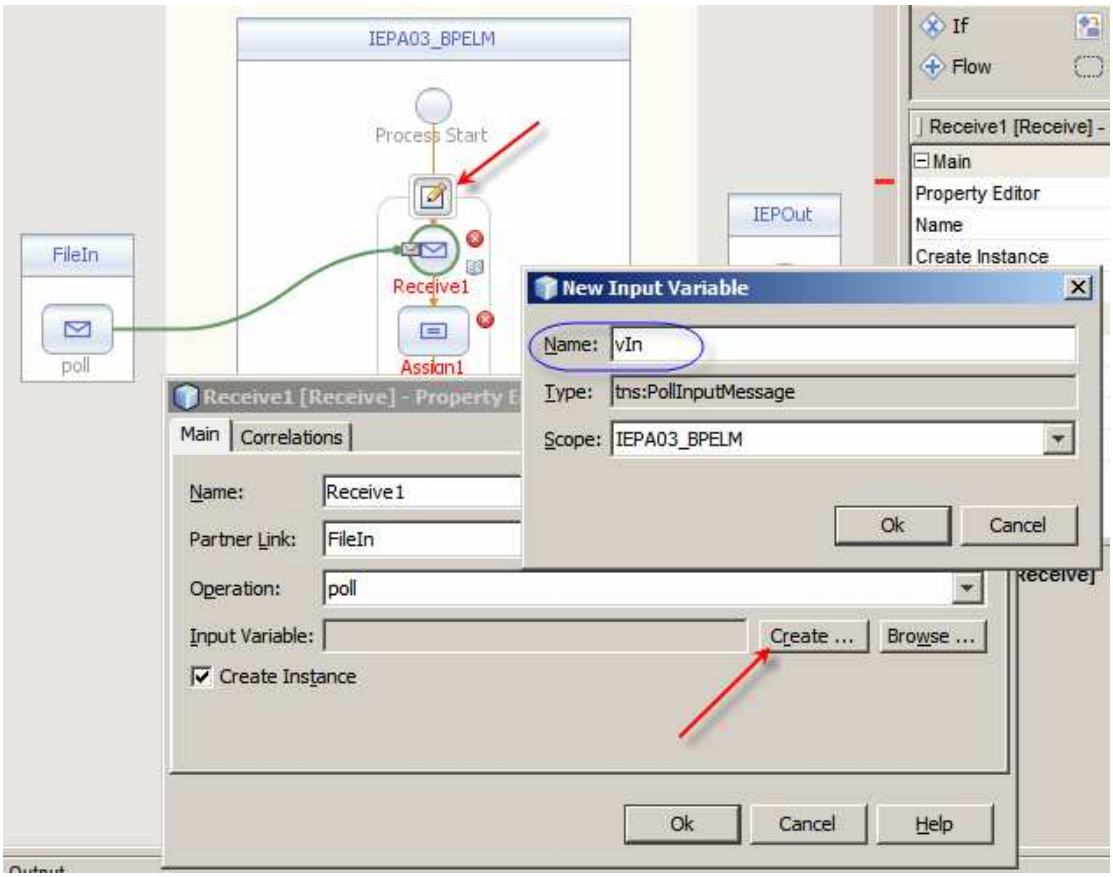
Drag receive, assign and invoke to the process scope.

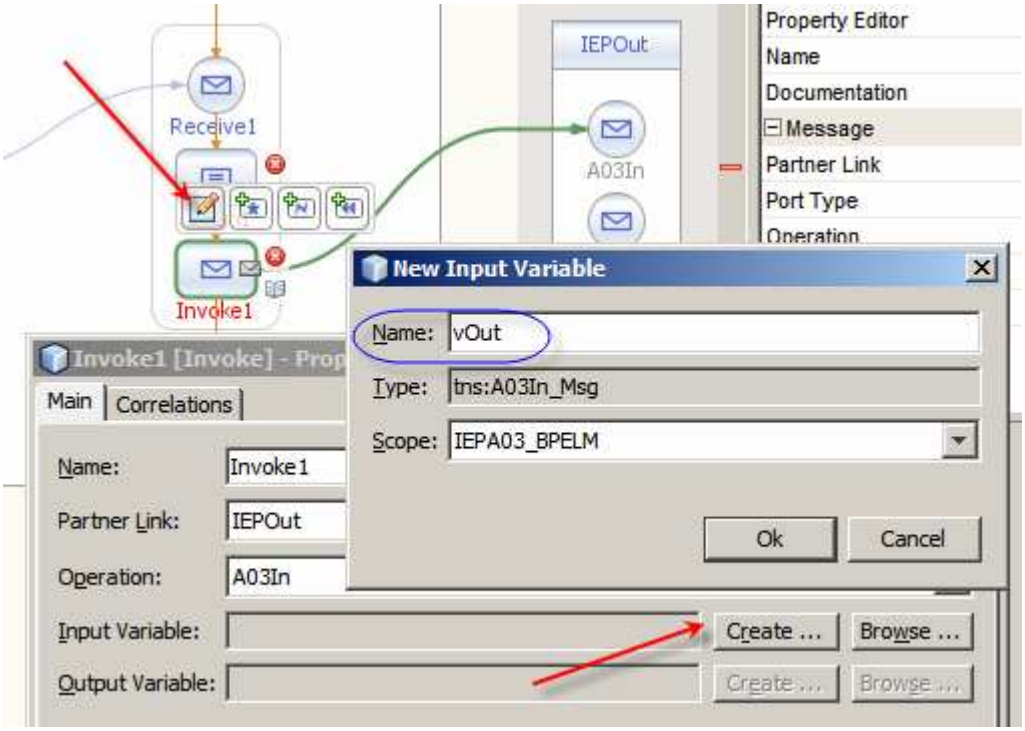


Connect Receive1 to FileIn and out Invoke1 to IEPOut A03In operation.

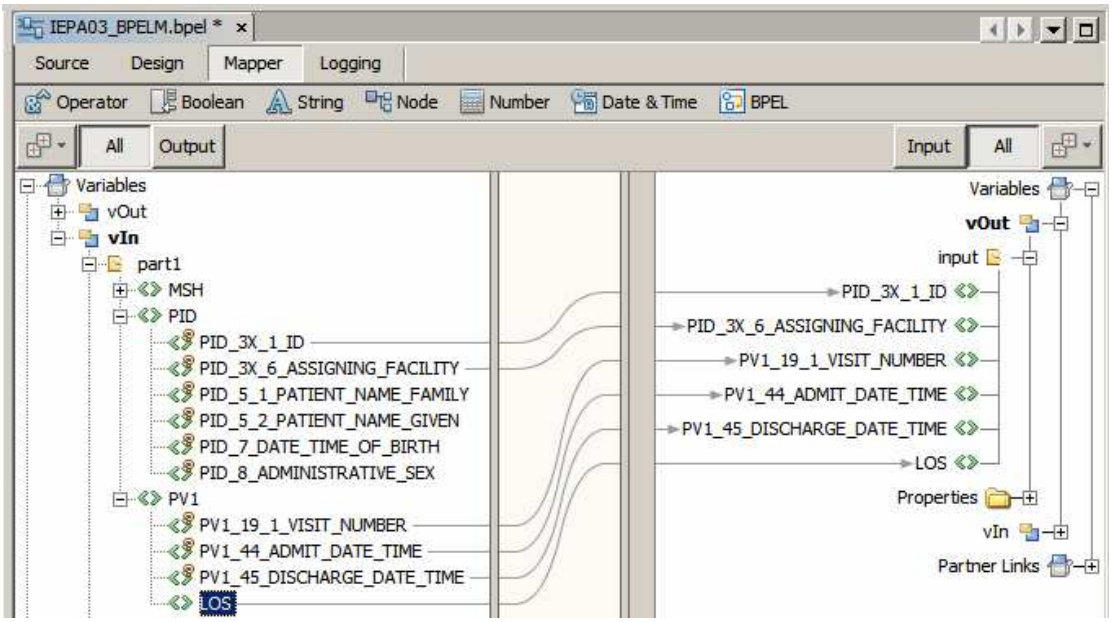


Add variables vIn for Receive1 and vOut for Invoke1.





Switch to Mapper mode and map nodes of vIn to corresponding nodes of vOut.



Build the project.

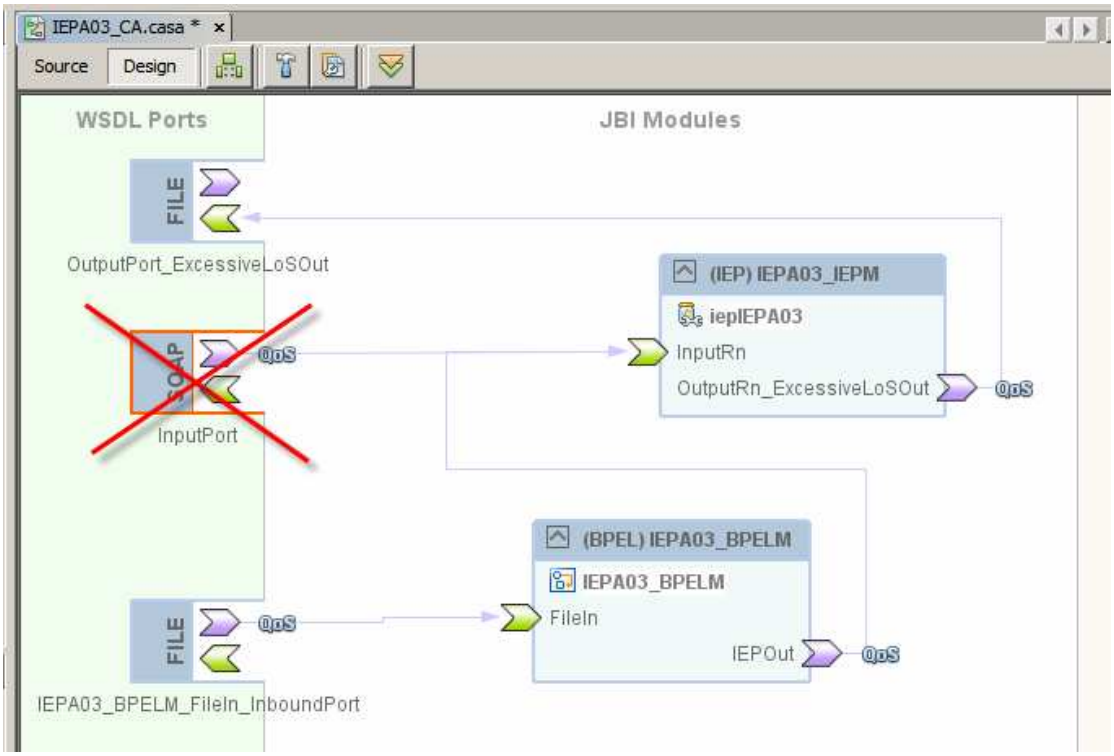
The BPEL module, which sends records to the IEP processor, is ready

Building and Deploying Composite Application

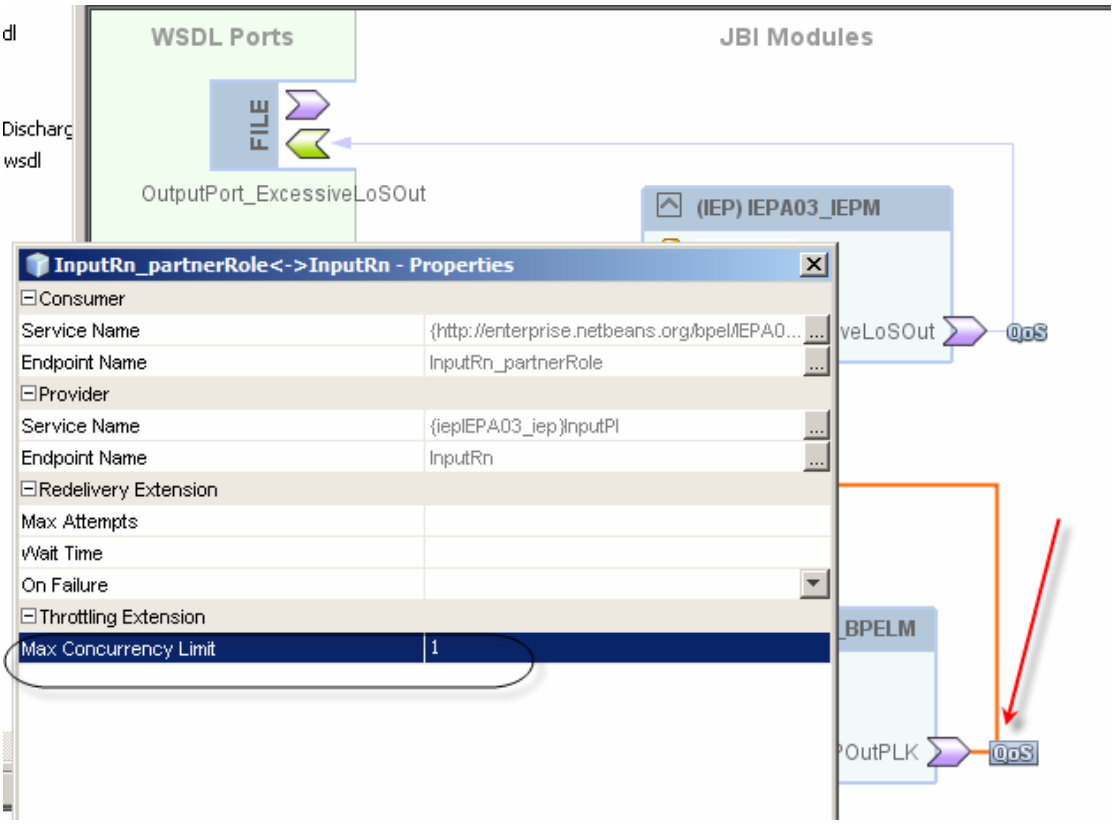
Create “New Project” -> “SOA” -> “Composite Application”, named IEPA03_CA.

Drag IEPA03_IEPM and IEPA03_BPELM modules onto the CASA map and build.

Once the composite application is built, delete the superfluous SOAP BC.



Edit link properties of the link between the IEP Module and the BPEL Module. Configure Max Concurrency Limit to 1



Build and Deploy the composite application.

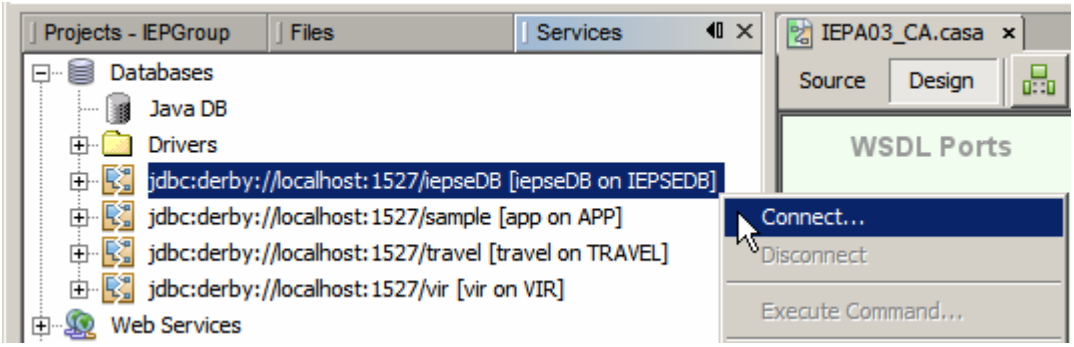
The runtime solution is now ready to process data

Exercising Solution

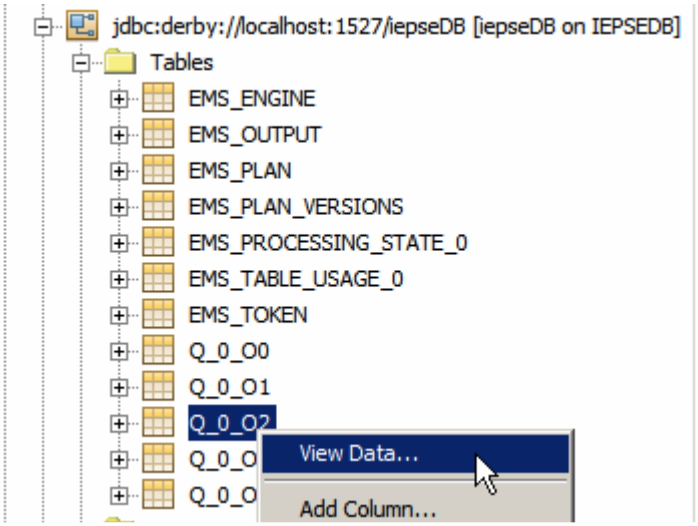
Submit IEP_output_1.dat by copying it from <project folder>/Combined/data/sources folder to <project folder>/Combined/data folder.

There will be nothing produced since the condition filters out the record.

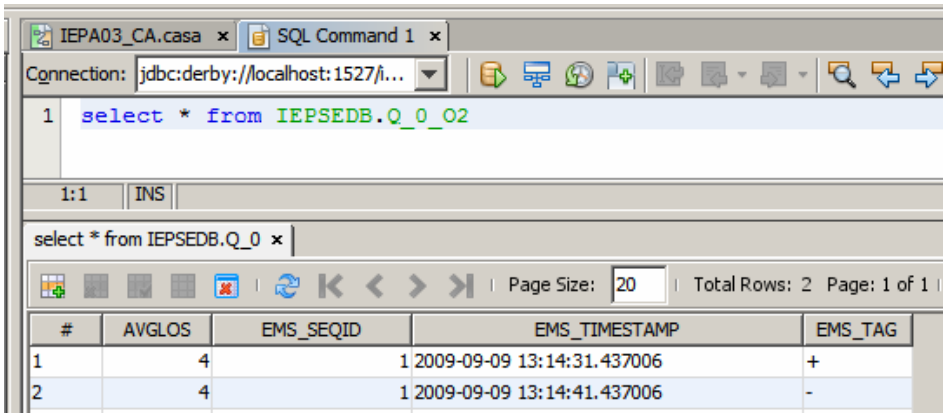
Switch to NetBean -> Services -> Databases. Right-click the iepseDB connection and choose Connect.



Expand Tables node, right-click Q_0_o2 table name and choose “View Data”.



Inspect data in table Q_0_O2.



Submit IEP_output_10.dat by copying it from <project folder>/Combined/data/sources folder to <project folder>/Combined/data folder.

Look for an output file, ExcessiveLoSOut.txt, in <project folder>/Combined/data.

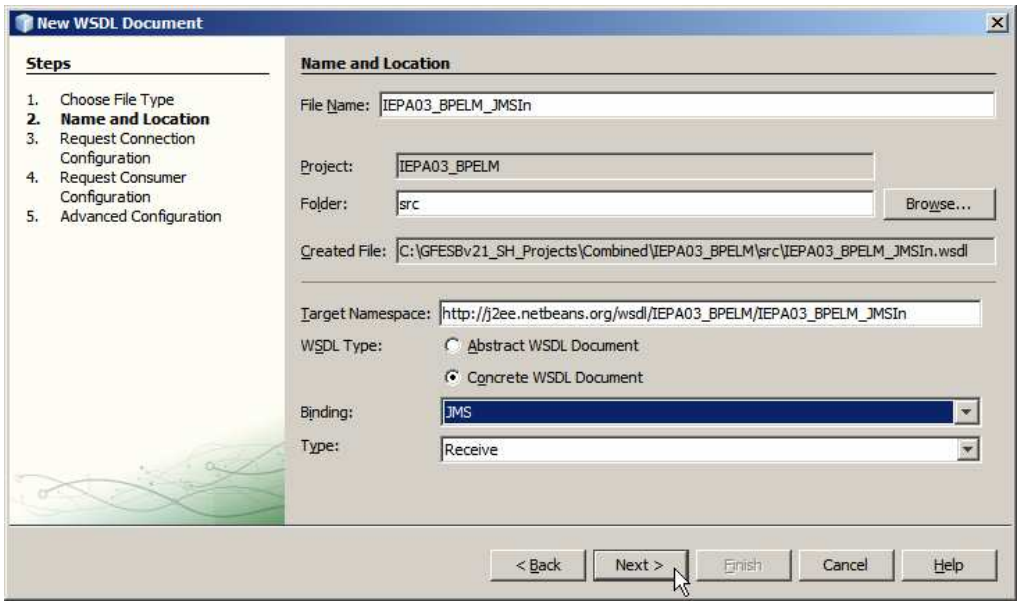
Of the 10 transactions submitted to the solution 2 had excessive length of stay, bot of 4 days where the average length pf stay for the transactions in the window at the time was 2 days.

Submit IEP_output_1000.dat by copying it from <project folder>/Combined/data/sources folder to <project folder>/Combined/data folder. 1000 transactions ought to take more then 10 seconds to process so there ought to be time to observe contents of tables Q_0_oX at different time during process execution. In particular, table Q_0_O1 (TBW10Seconds) shows transactions in the time window and Q_0_O3 shows transactions with excessive length of stay (ExcessiveLoS).

Adding JMS Feeder to the Solution

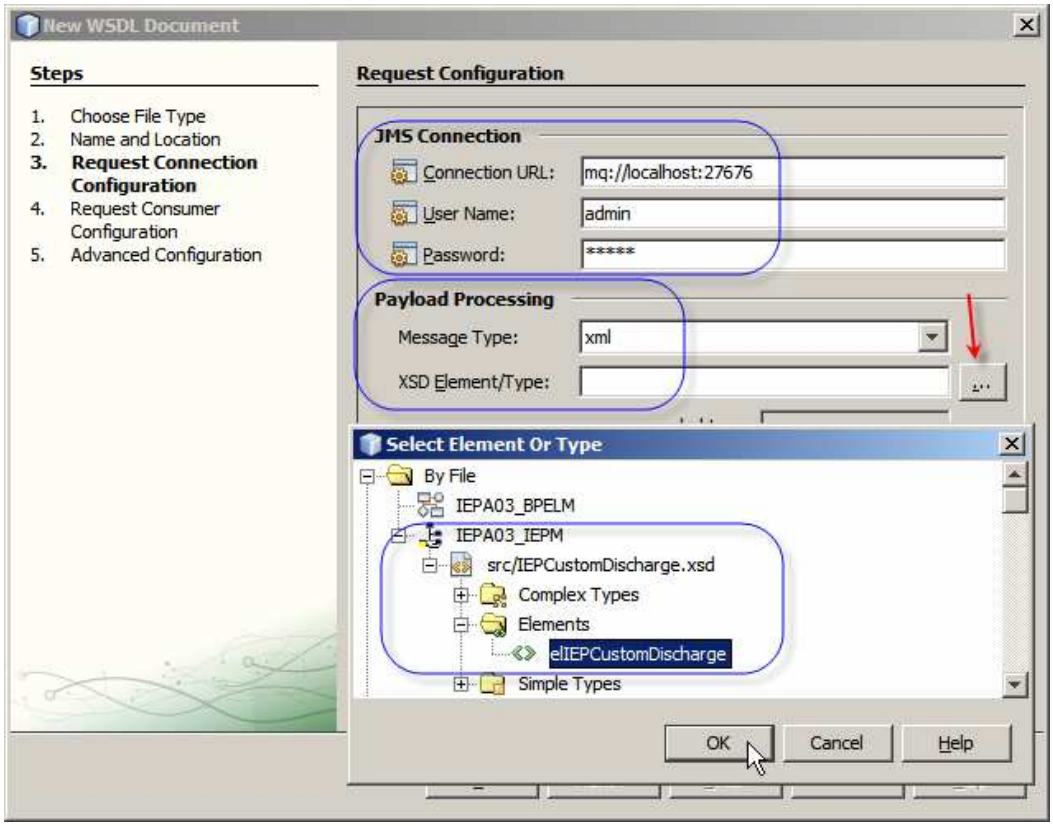
What we are about to implement is called a “Service Activator” pattern - multiple methods of invoking the same service. We already have a File BC invoking the BPEL process. We will now add a JMS BC which will be used to send messages to the BPEL process as well.

Open the IEPA03_BPELM project. Right-click Process Files. Choose “New” -> “WSDL Document”/ Name the WSDL IEPA03_BPELM_JMSIn. Make it a Concrete WSDL, using JMS Binding, of Type Receive.



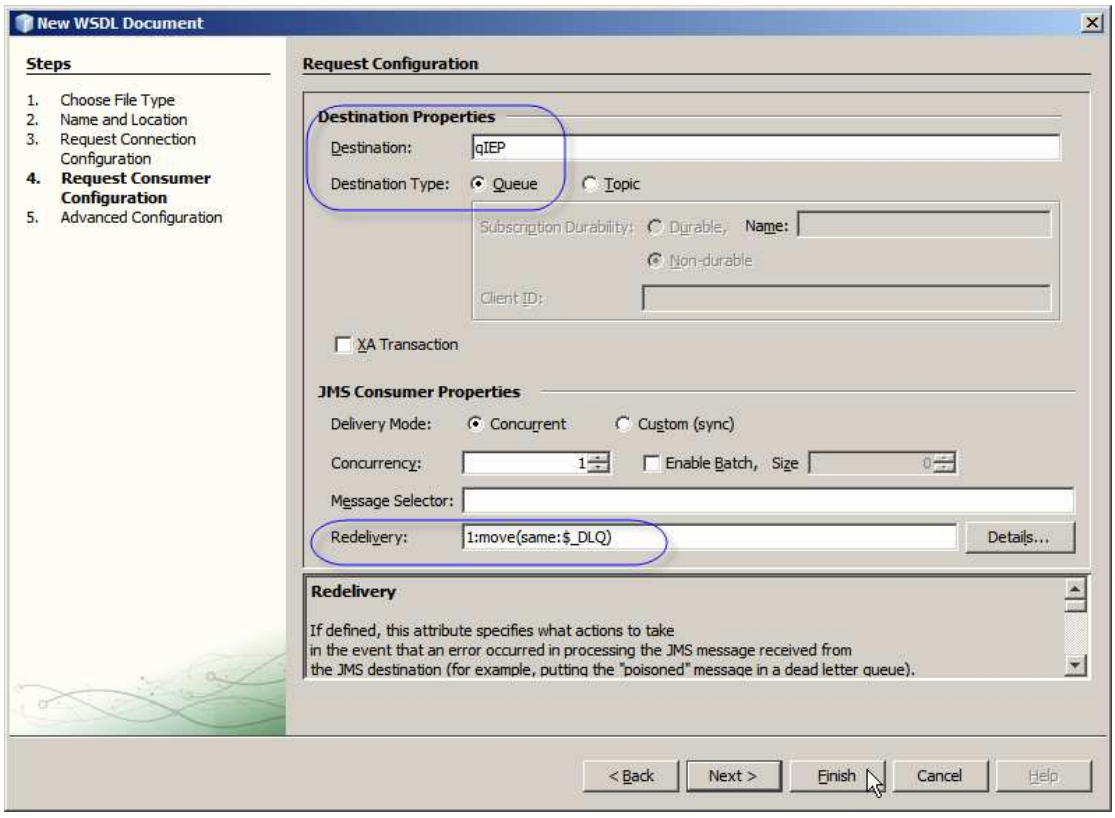
Click Next.

Set Connection URL: mq://<yourMQhost>:<yourMQport>, User Name: admin, Password: admin, Message Type: xml, XSD Element/Type: elIEPCustomDischarge from the IEPA03_IEPM project.



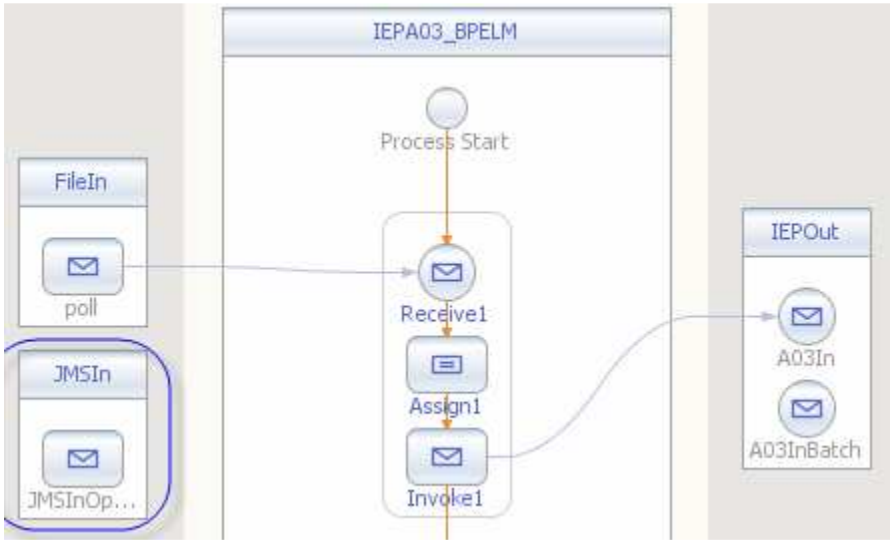
Click Next.

Set Destination: qIEP, Destination Type: Queue, Redelivery: 1:move(same:$_DLQ).

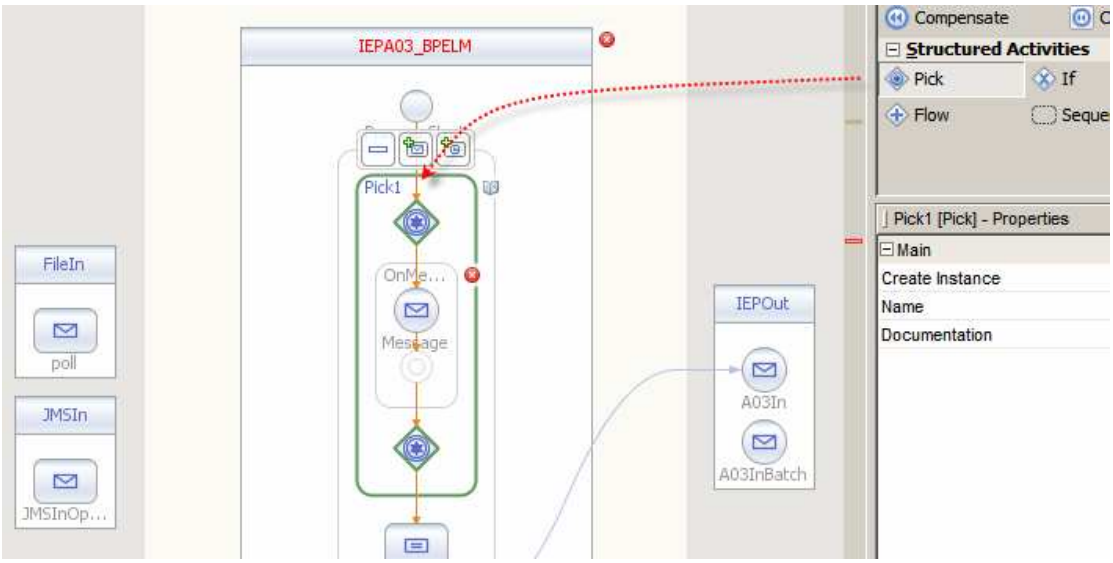


Click Finish to complete the configuration wizard.

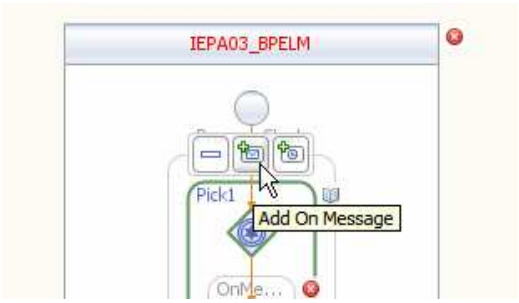
Open the IEPA03_BPELM.bpel process in the editor. Drag the new IEPA03_BPELM_JMSIn WSDL onto the target market at the left hand swim line. Name the new partner link JMSIn.



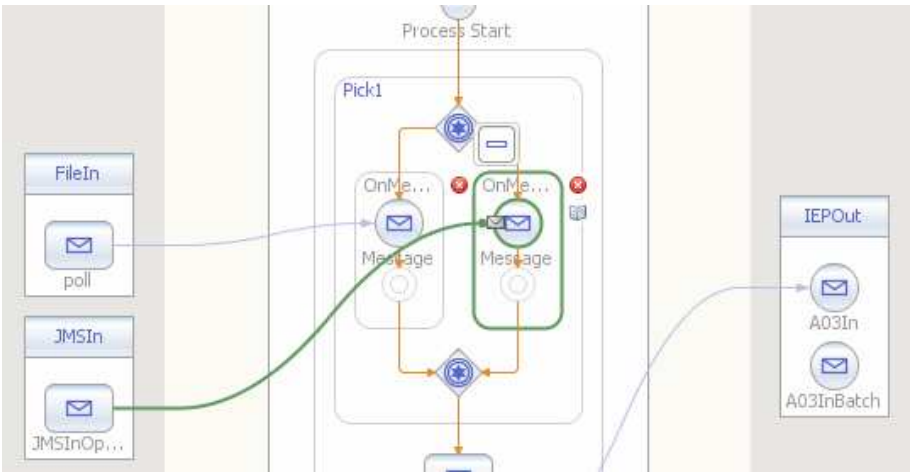
Delete Receive1 activity. Drag the “Structured Activities” -> “Pick” activity from the Palette onto the target marker inside the process scope to replace the Receive1 activity you just deleted.



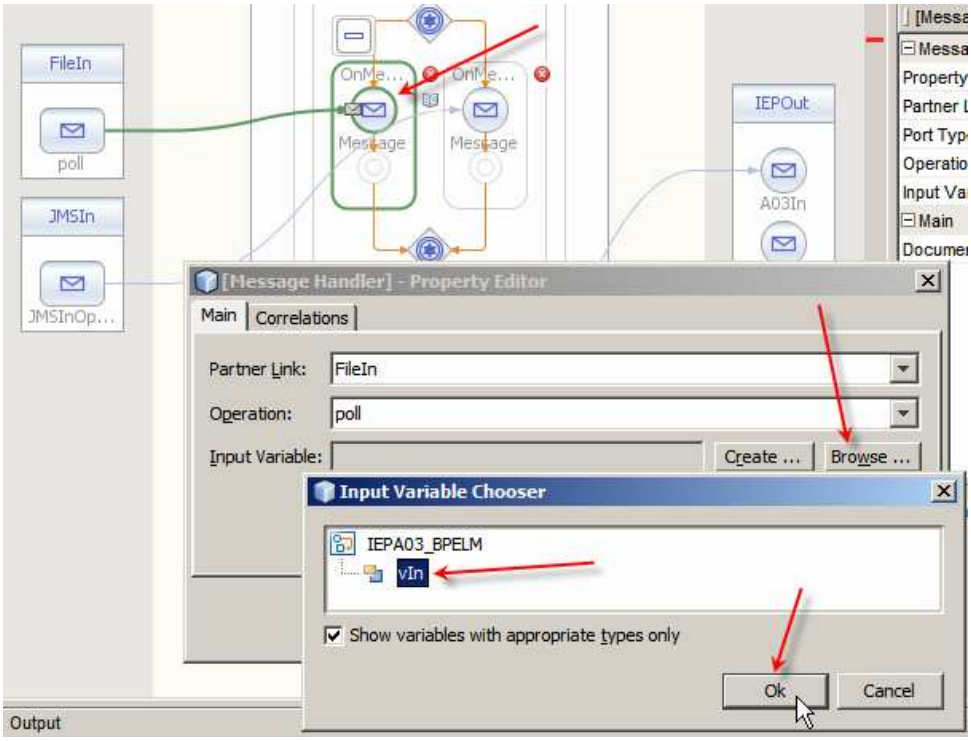
Select the Pick1 scope. Click the “Add on Message” icon.



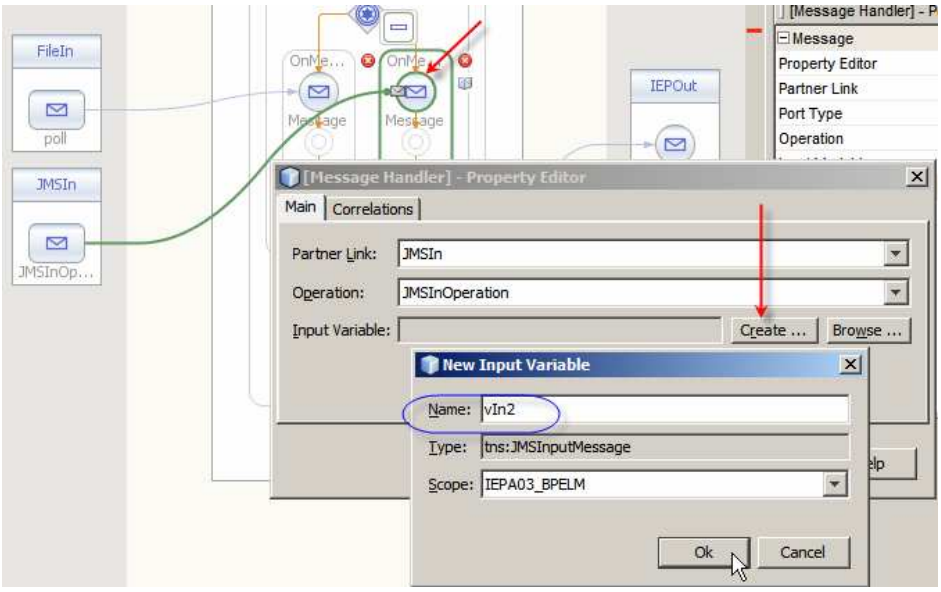
Connect the first onMessage activity to the FileIn partner link and the second onMessage activity to the JMSIn partner link.



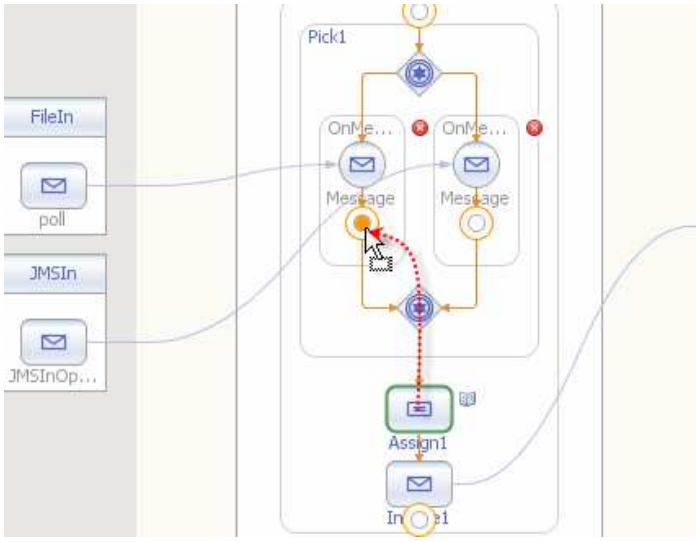
Double-click the first onMessage activity, click Browse and choose the existing vIn variable.



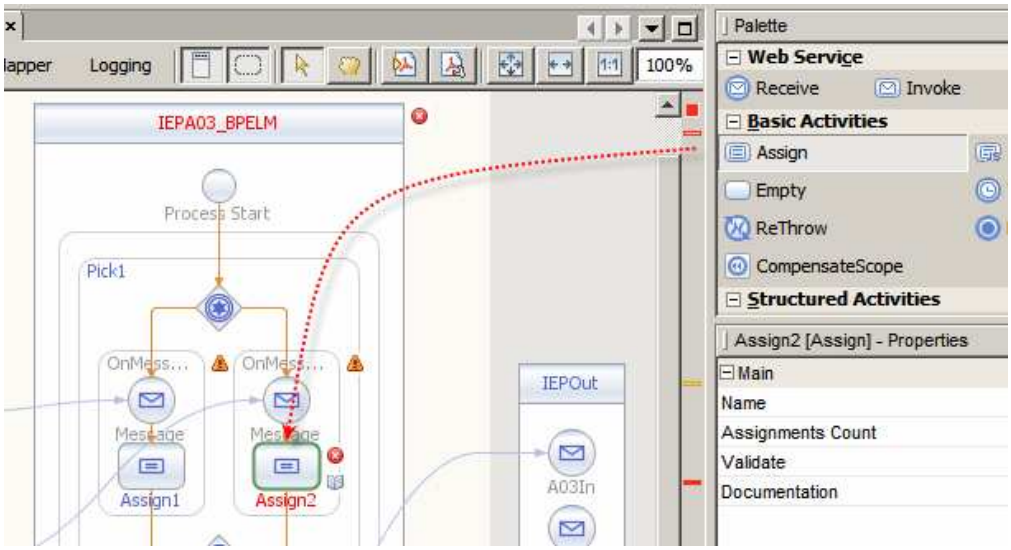
Double-click the second onMessage activity, click Create and create a new vIn2 variable.



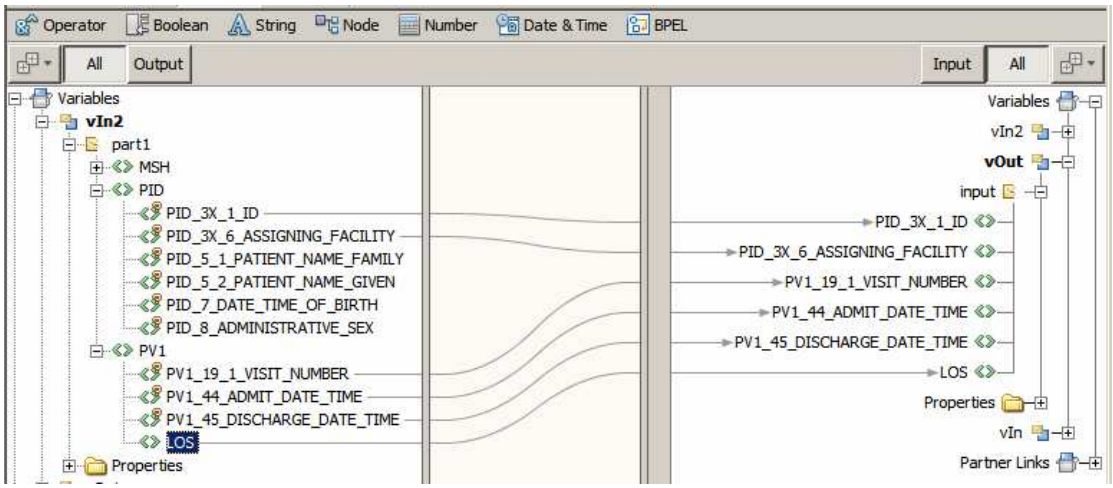
Drag the Assign1 activity onto the target marker below the first onMessage activity inside its scope.



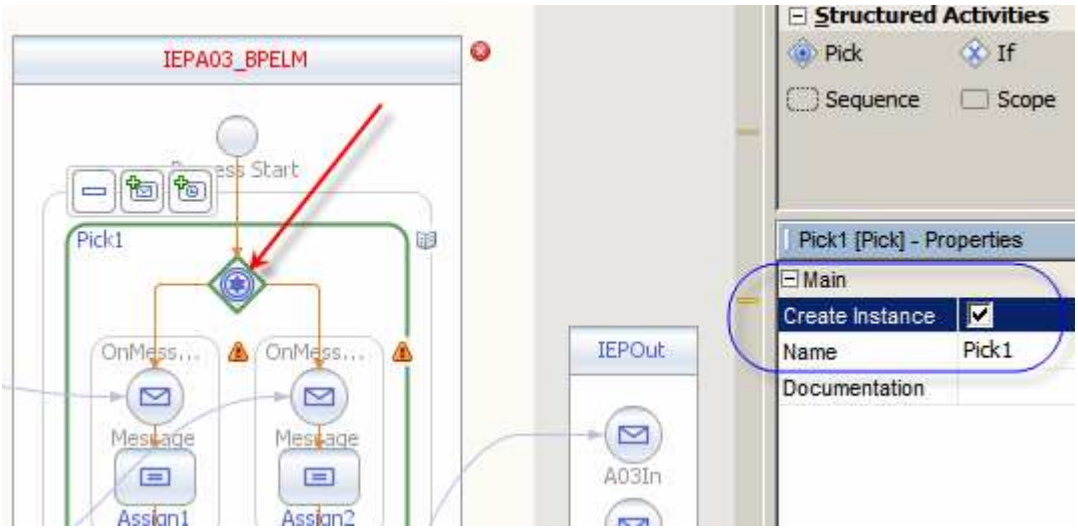
Add Assign activity below the second onMessage.



Double-click the Assign2 activity and map nodes of vIn2 to corresponding nodes of vOut as shown.

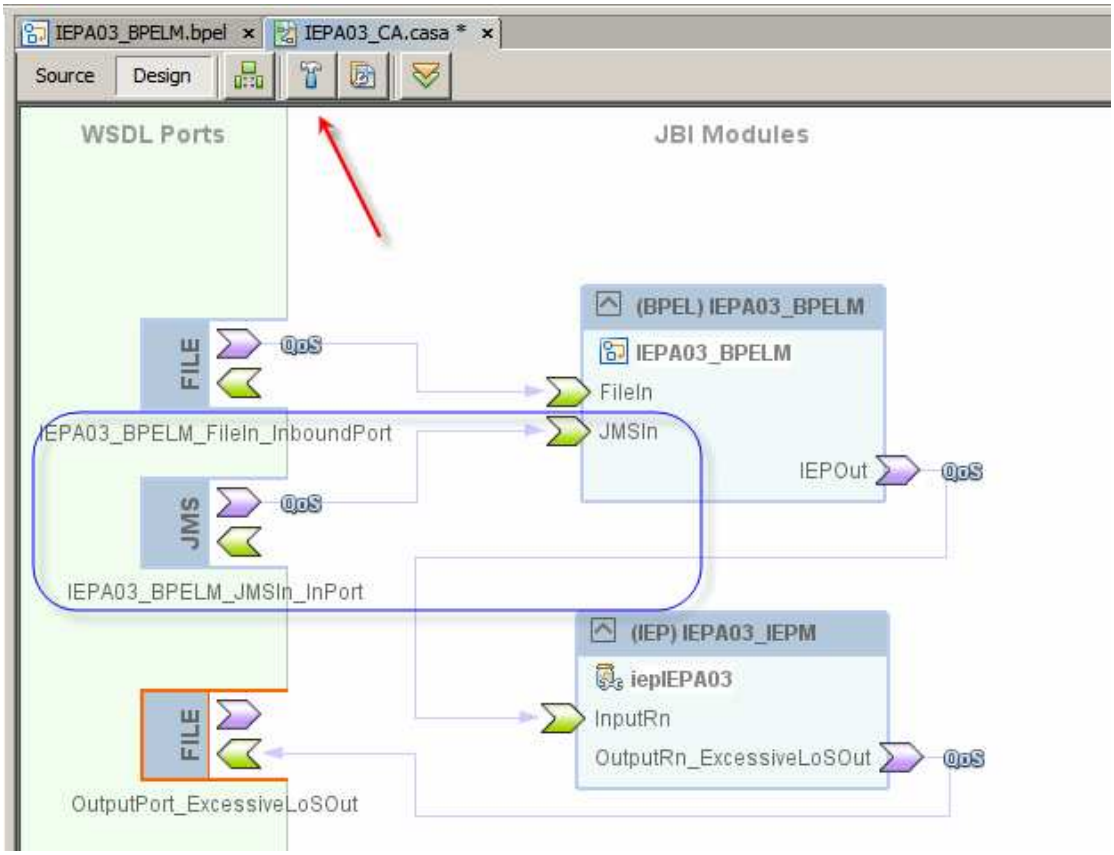


Click the Pick1 activity and check the “Create Instance” checkbox.



Save and Build the project.

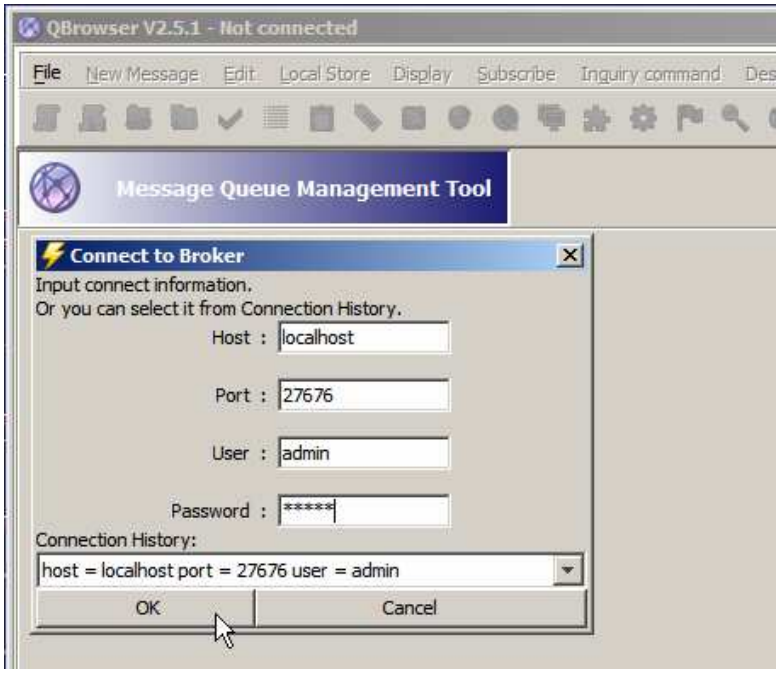
Open the IEPA03_CA -> “Service Assembly” and click Build.

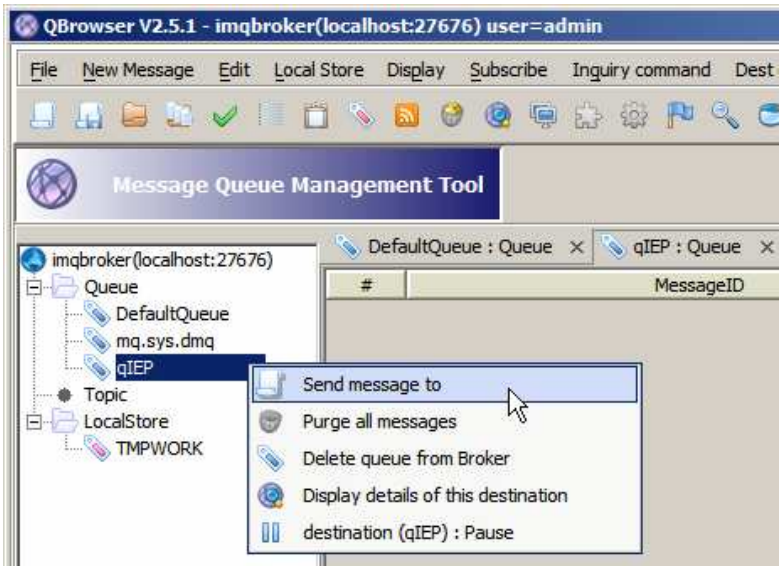


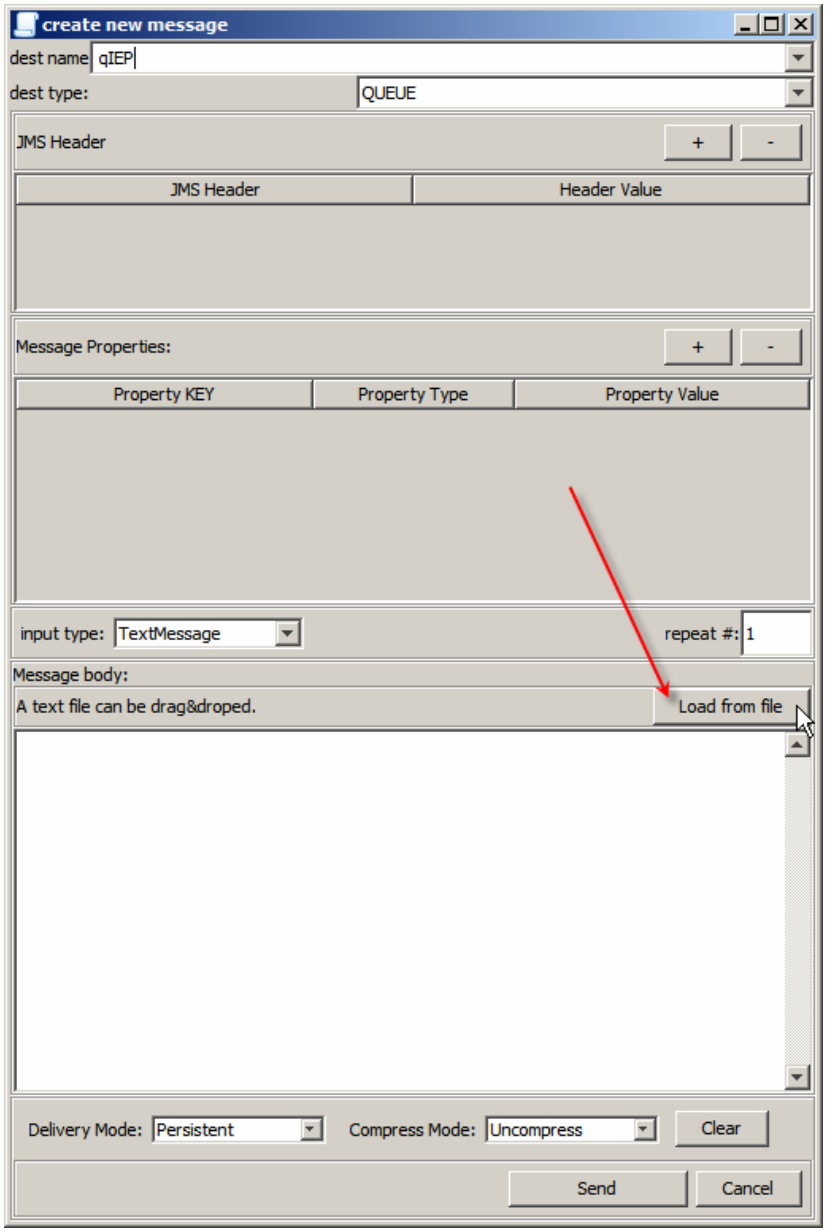
Deploy the project.

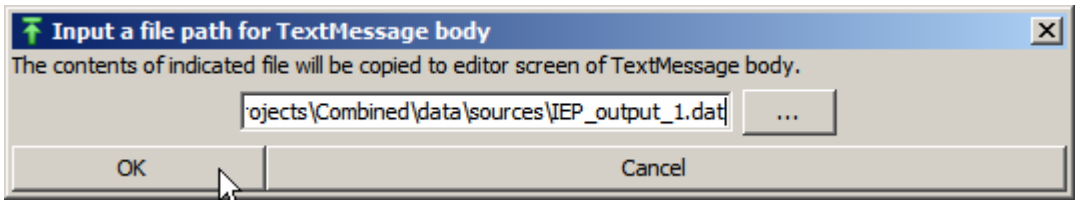
Use a tool like QBorwser (http://sourceforge.net/projects/qbrowserv2/) to connect to the JMS implementation and submit one or more message from the <project root>/Combi

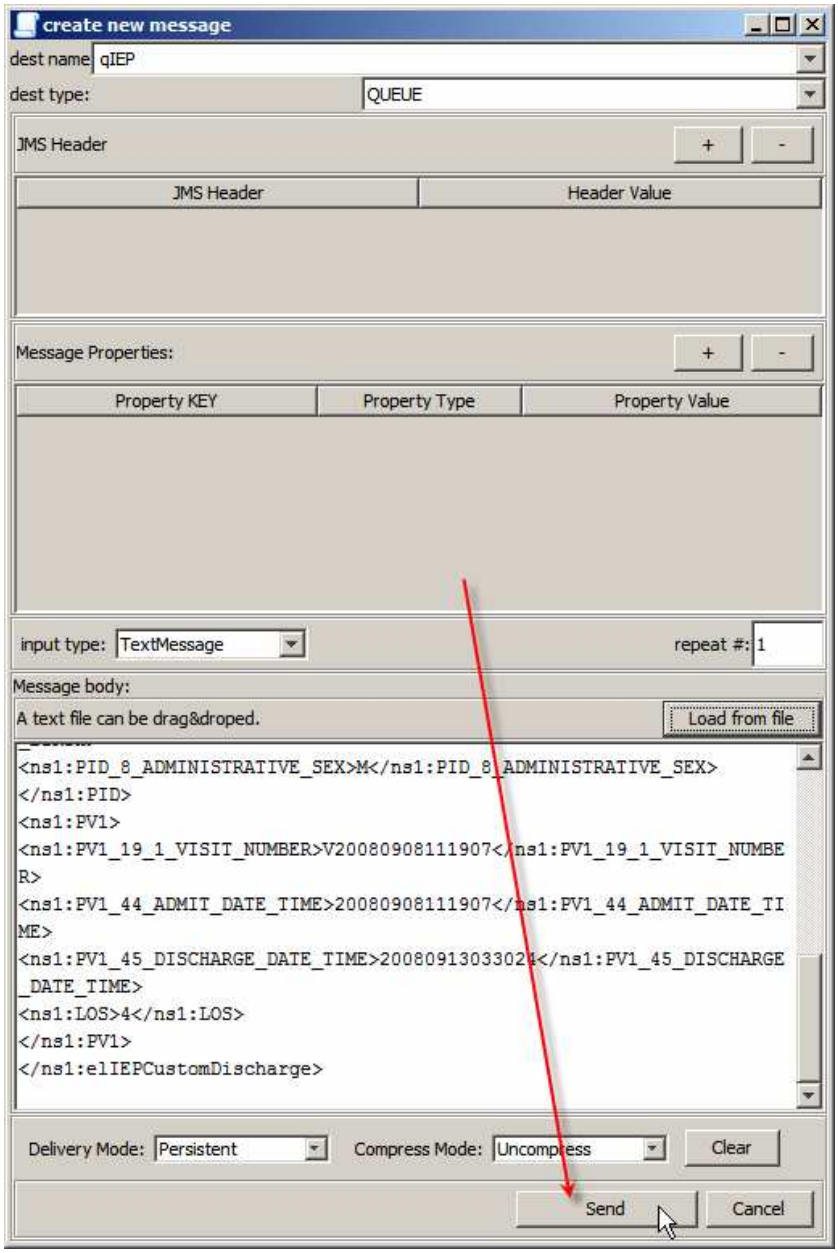
ned/data/sources/IEPOutput_10.dat to the JMS Queue qIEP. Observe server.log to make sure that solution still works.

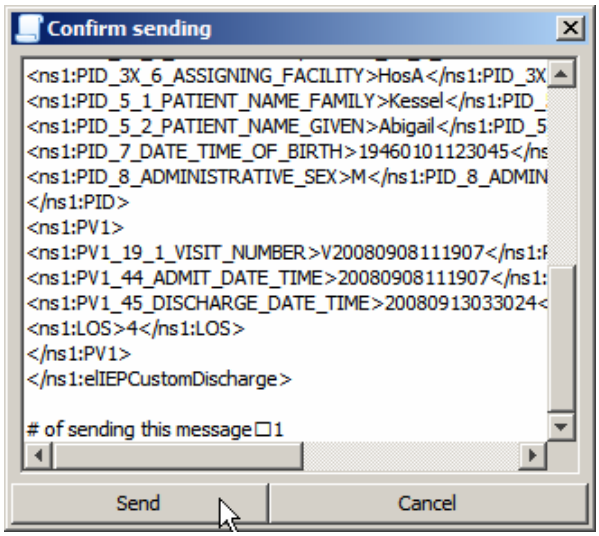


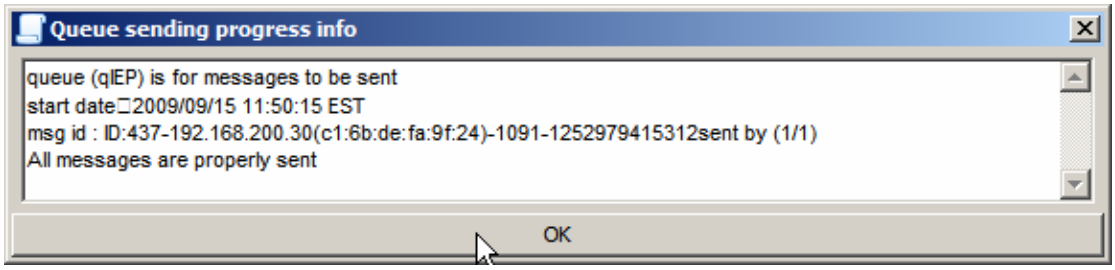



create new message
dest name: qIEP
dest type: QUEUE
JMS Header
+
-
JMS Header
Header Value
Message Properties:
+
-
Property KEY
Property Type
Property Value
input type: TextMessage
repeat #: 1
Message body:
A text file can be drag&droped.
Load from file
Delivery Mode: Persistent
Compress Mode: Uncompress
Clear
Send
Cancel


Input a file path for TextMessage body
The contents of indicated file will be copied to editor screen of TextMessage body.
ojects\Combined\data\sources\IEP_output_1.dat
...
OK
Cancel


create new message
dest name: qIEP
dest type: QUEUE
JMS Header
+
-
JMS Header
Header Value
Message Properties:
+
-
Property KEY
Property Type
Property Value
input type: TextMessage
repeat #: 1
Message body:
A text file can be drag&droped.
Load from file
<ns1:PID_8_ADMINISTRATIVE_SEX>M</ns1:PID_8_ADMINISTRATIVE_SEX>
</ns1:PID>
<ns1:PV1>
<ns1:PV1_19_1_VISIT_NUMBER>V20080908111907</ns1:PV1_19_1_VISIT_NUMBE
R>
<ns1:PV1_44_ADMIT_DATE_TIME>20080908111907</ns1:PV1_44_ADMIT_DATE_TI
ME>
<ns1:PV1_45_DISCHARGE_DATE_TIME>20080913033024</ns1:PV1_45_DISCHARGE
_DATE_TIME>
<ns1:LOS>4</ns1:LOS>
</ns1:PV1>
</ns1:elIEPCustomDischarge>
Delivery Mode: Persistent
Compress Mode: Uncompress
Clear
Send
Cancel


Confirm sending
<ns1:PID_3X_6_ASSIGNING_FACILITY>HosA</ns1:PID_3X
<ns1:PID_5_1_PATIENT_NAME_FAMILY>Kessel</ns1:PID_
<ns1:PID_5_2_PATIENT_NAME_GIVEN>Abigail</ns1:PID_5
<ns1:PID_7_DATE_TIME_OF_BIRTH>19460101123045</ns
<ns1:PID_8_ADMINISTRATIVE_SEX>M</ns1:PID_8_ADMIN
</ns1:PID>
<ns1:PV1>
<ns1:PV1_19_1_VISIT_NUMBER>V20080908111907</ns1:
<ns1:PV1_44_ADMIT_DATE_TIME>20080908111907</ns1:
<ns1:PV1_45_DISCHARGE_DATE_TIME>20080913033024<
<ns1:LOS>4</ns1:LOS>
</ns1:PV1>
</ns1:elIEPCustomDischarge>
# of sending this message 1
Send
Cancel

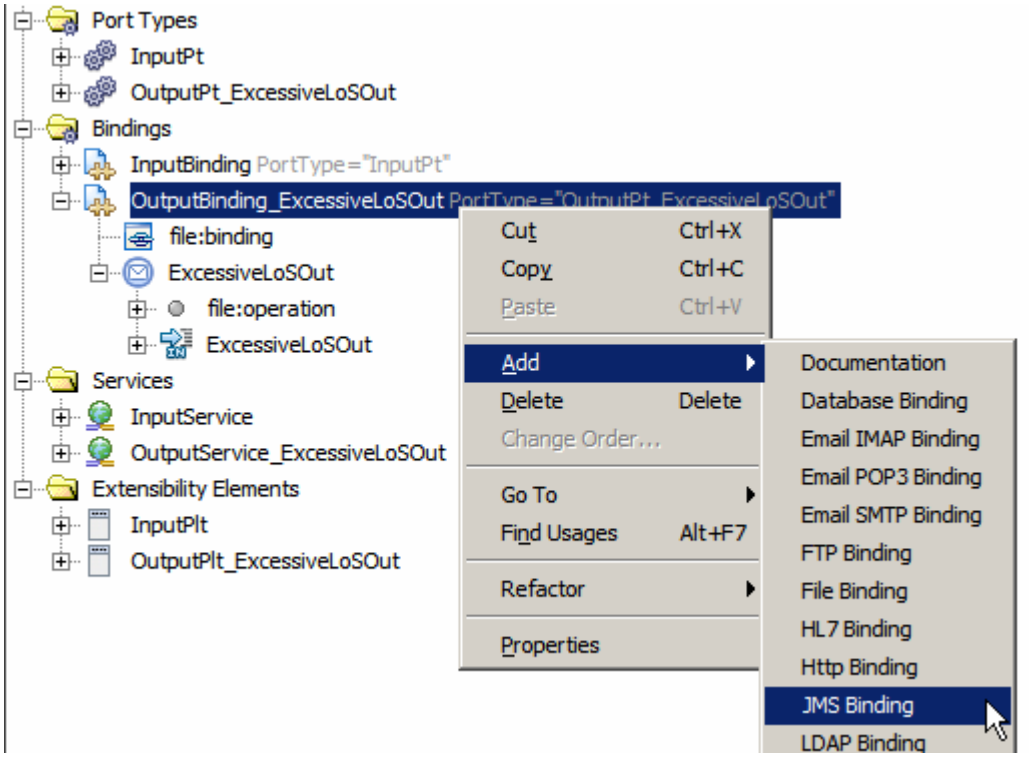


The project still works and can now receive messages from files and over JMS.

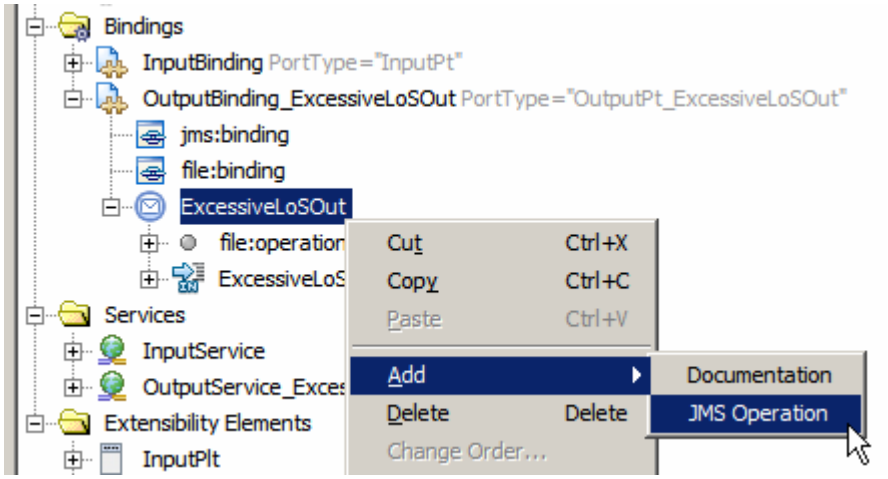
Adding JMS Notification Receiver to the Solution

Currently notification messages, these where the length of stay exceeds average by more then 1 ½ times, are written to a file. This is not particularly useful as there may be a significant delay between the time a notification is written and the time somebody gets informed. To give us the flexibility to add an automated notificaitn mechanism we will modify the IEP process to send notification messages to a JSM Queue instead of writing them to a file.

Open IEPA03_IEPM -> Processor Files -> iepIEPA03.wsdl in the WSDL editor. Expand Bindings node. Right-click on the OutputBinding_ExcessiveLoSOut. Choose Add -> JMS Binding.



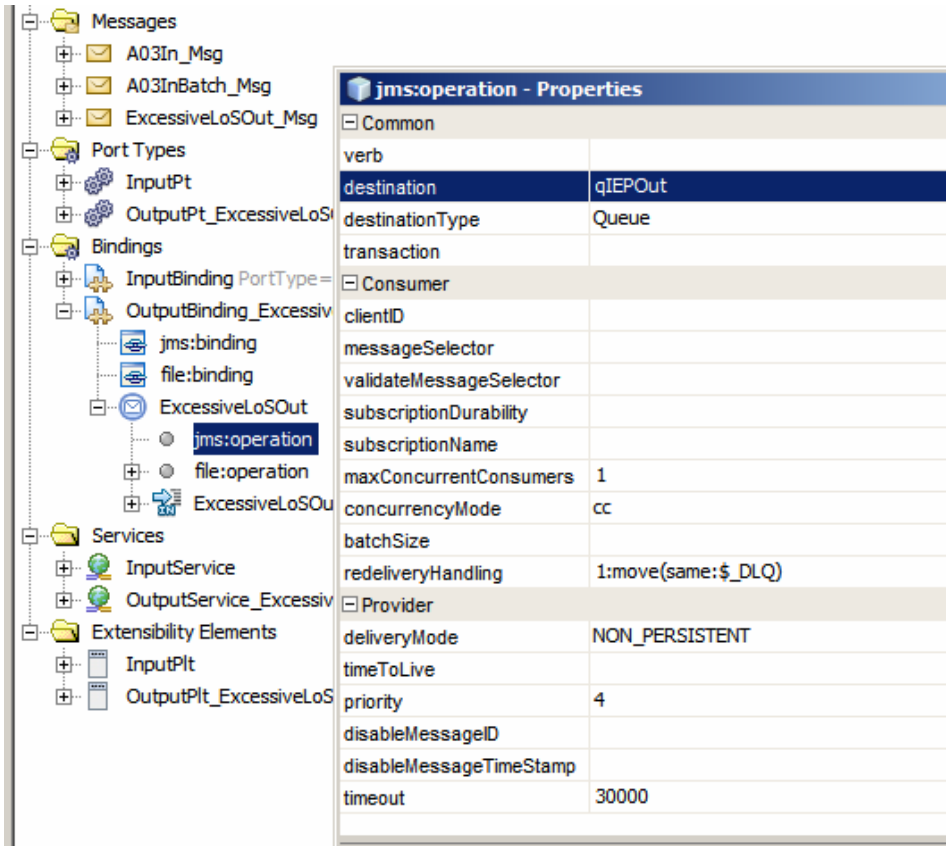
Right-click the ExcessiveLoSOut operation node. Choose Add -> JMS Operation.



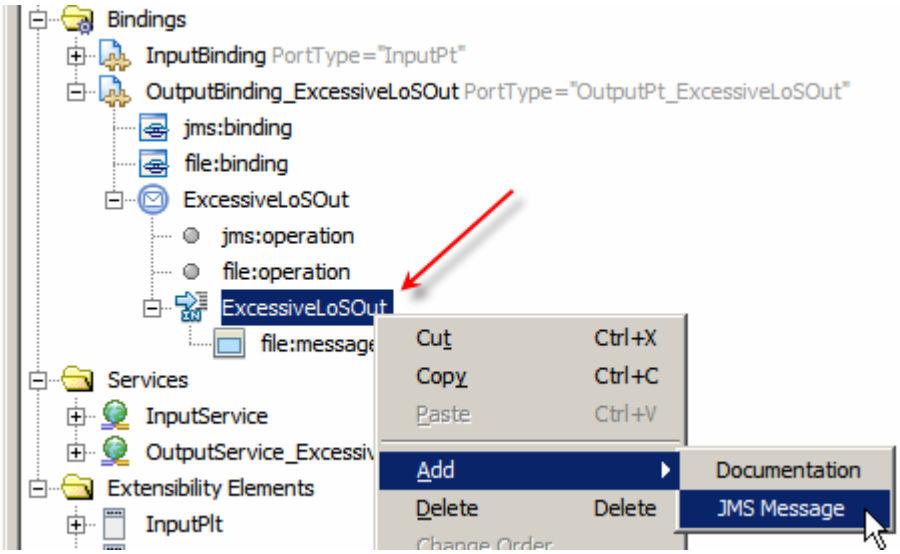
Right-click the new jms:operation node and choose Porperties.

Configure key properties:

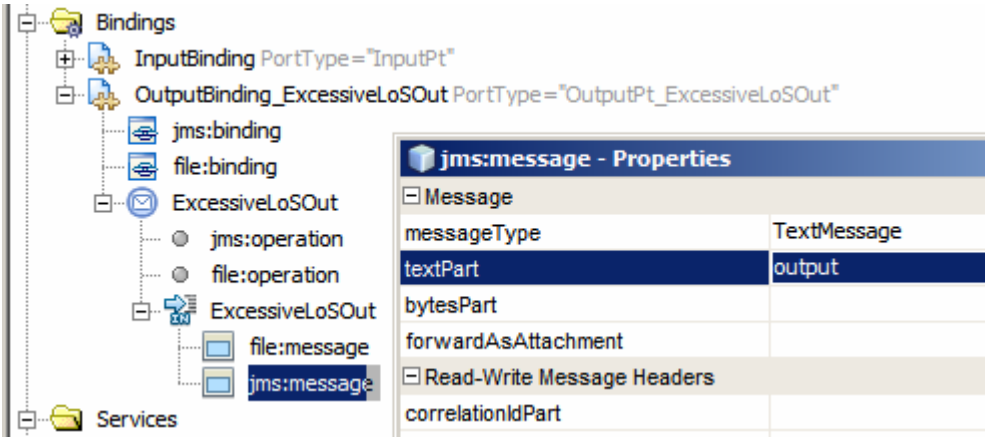
- destination: qIEPOut
- destinationType: Queue
- maxConcurrentConsumers: 1
- concurrencyMode: cc
- redeliveryHandling: 1:move(same:$_DLQ)
- deliveryMode: NON_PERSISTENT
- priority: 4
- timeout: 30000



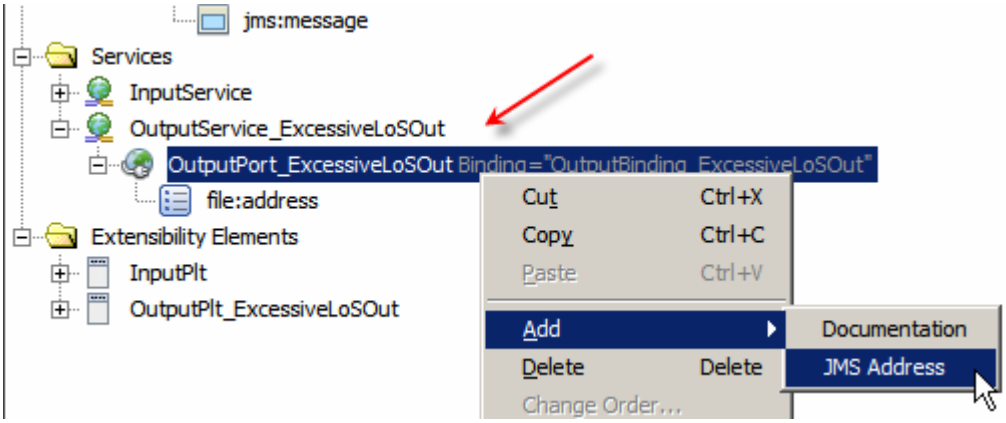
Right-click ExcessiveLoSOut message. Choose Add -> JMS Message.



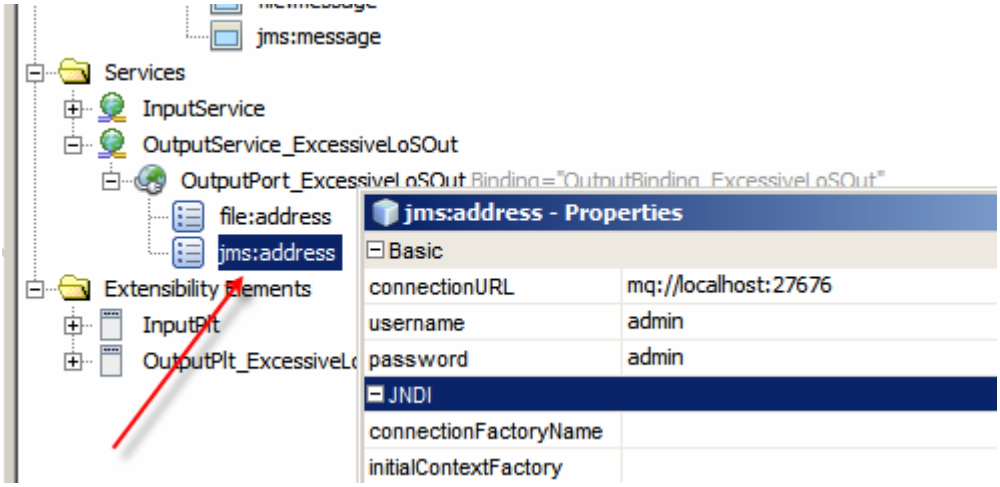
Right-click the new jms:message. Choose Proeprties. Configure properties: messageType: TextMessage, textPart: output.



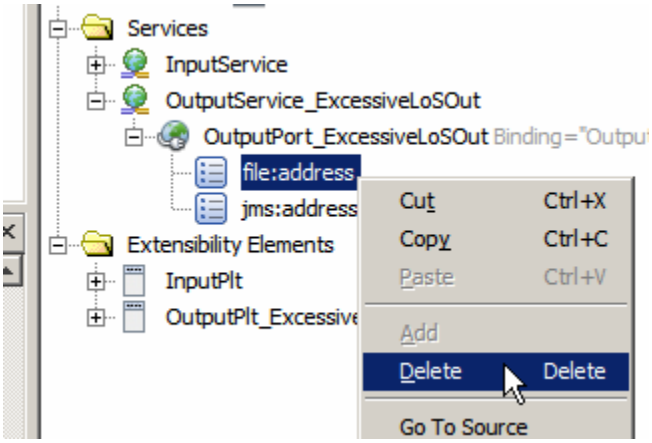
Expand Services node all the way to OutputService_ExcessiveLoSOut. Right-click OutputPort_ExcessiveLoSOut node. Choose Add -> JMS Address.



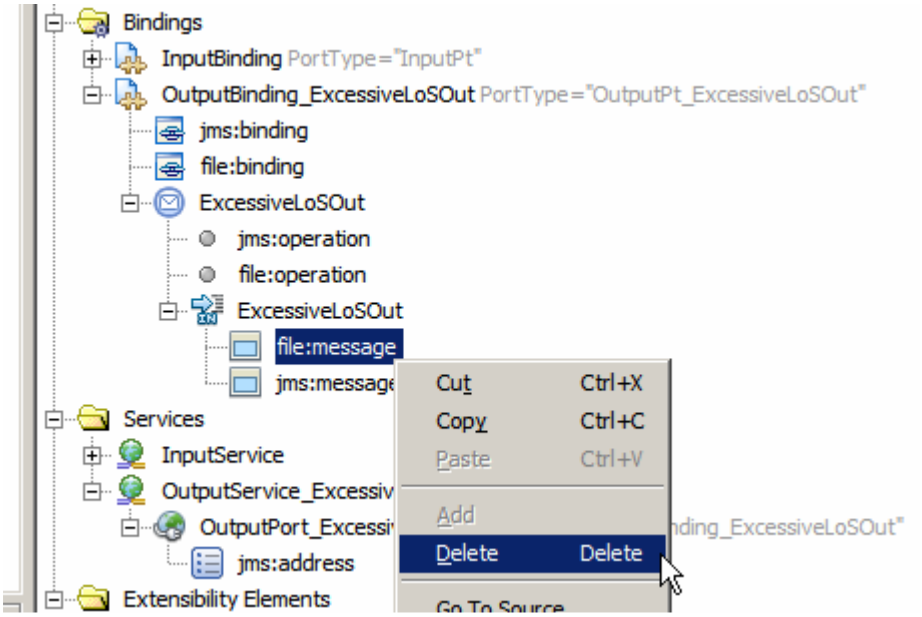
Right-click the new jms:address node. Choose Porperties. Configure properties as appropriate for your environment.



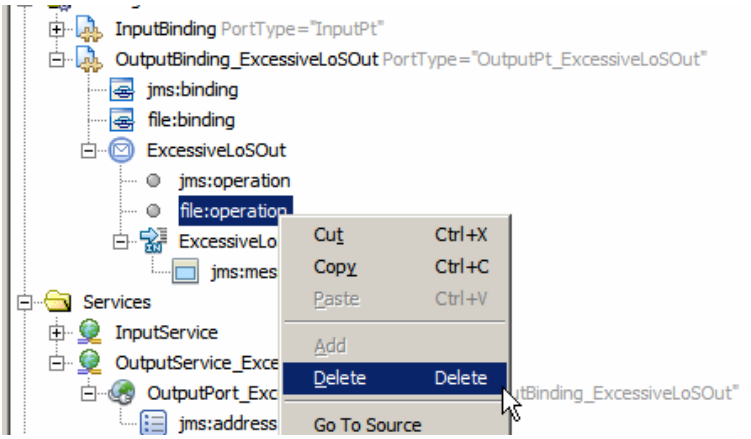
Right-click file:address and choose Delete.



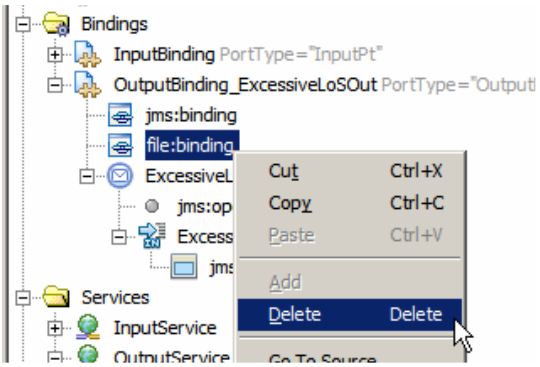
Right-click file:message and choose Delete.



Right-click file:operation and choose Delete.



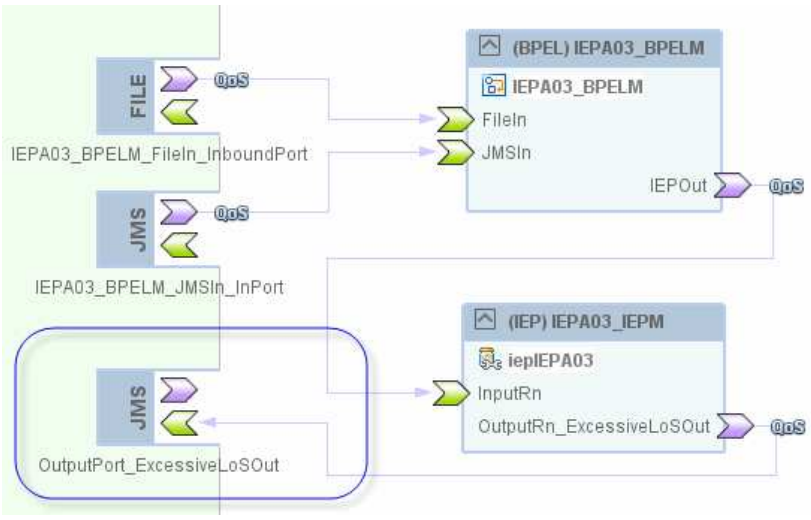
Right-click file:binding and choose Delete.



Save the WSDL and Build the IEP project.

Open the IEPA03_CA -> Service Assembly and Build.

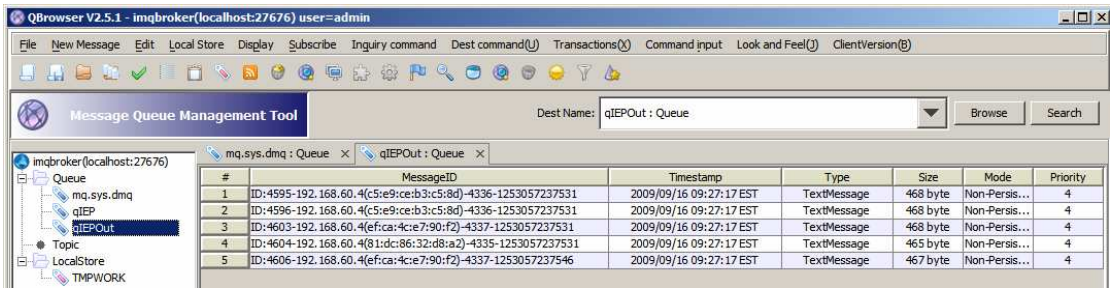
Note that the IEP output binding changed from File BC to JMS BC.

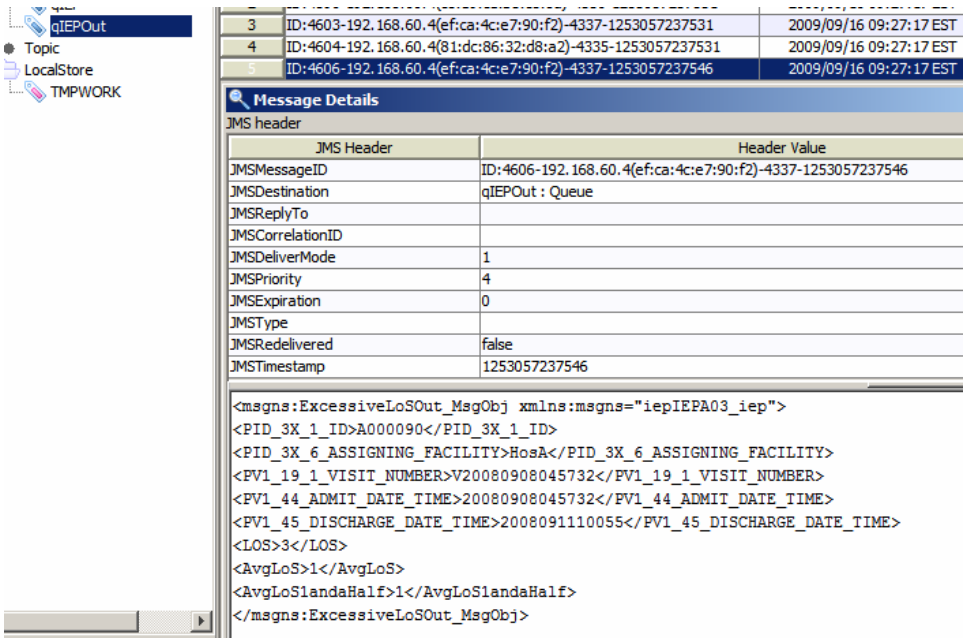


Set the maxConcurrencyLimit property of the connection between the BPEL Process and the IEP Processor to 1 again.

Build and Deploy the project.

Submit the 10-record file, IEP_output_10.dat, and observe the appearance of the qIEPOut and the appearance of messages in that Queue. There may be a delay of a few seconds as the IEP batches output stream to optimize performance.





The IEP messages are now queueud to a JMS Queue instead of being written to a file.

Adding Notification Sender

Now that we have the notifications messages in a JMS Queue we can develop a solution which will process these notifications. In this case we will take each message and send it, using electronic mail, to a pre-configured recipient.

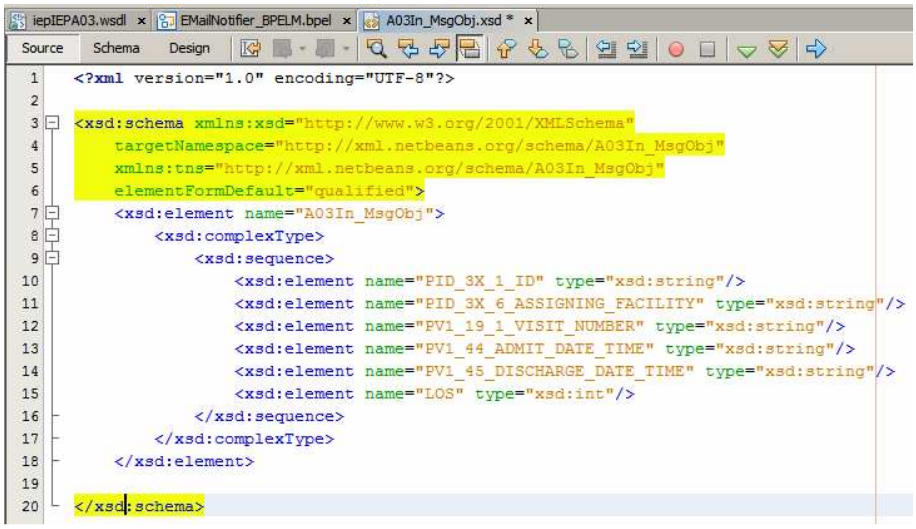
Right-click anywhere in the empty area of the project explorer and choose New Project -> SOA -> BPEL Module. Name this BPEL Module EMailNotifier_BPELM.

Create a “New” -> “XML Schema”, named A03In_MsgObj. When the XSD opens in the editor switch to the Source view.

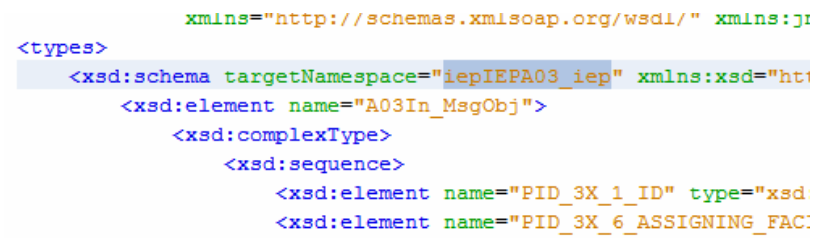
Open the “IEPA03_IEPM” -> “Processor Files” -> “iepIEPA03.wsdl” WSDL. Switch to the Source view, select the element named A03In_MsgObj and copy to the clipboard.

```
<xsd:schema targetNamespace="iepIEPA03_iep" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="A03In_MsgObj">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="PID_3X_1_ID" type="xsd:string"/>
                <xsd:element name="PID_3X_6_ASSIGNING_FACILITY" type="xsd:string"/>
                <xsd:element name="PV1_19_1_VISIT_NUMBER" type="xsd:string"/>
                <xsd:element name="PV1_44_ADMIT_DATE_TIME" type="xsd:string"/>
                <xsd:element name="PV1_45_DISCHARGE_DATE_TIME" type="xsd:string"/>
                <xsd:element name="LOS" type="xsd:int"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="A03InBatch_MsgObj">
```

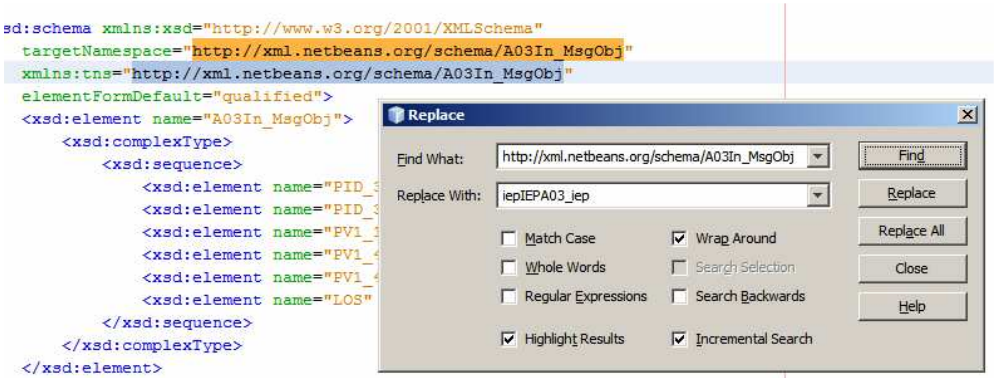
Switch to the A03In_MsgObj.xsd and paste inside the <schema …>…</schema> tags. Right-click inside the source and choose Format.



Switch back to the iepIEPA03.wsdl, select the value of the targetNamespace property and copy it to the clipboard.



Switch back to the A03In_MsgObj.xsd and replace all occurrences of the targetNamespace and xmlns:tns attribute values with the copied text.



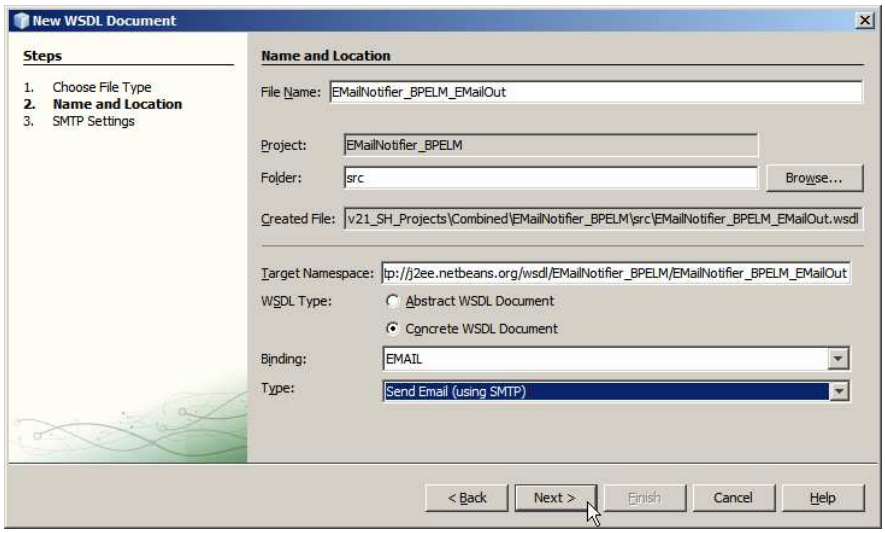
```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="iepIEPA03_iep"
    xmlns:tns="iepIEPA03_iep"
    elementFormDefault="qualified">
    <xsd:element name="A03In_MsgObj">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="PID_3X_1_ID" type="xsd:string"/>
                <xsd:element name="PID_3X_6_ASSIGNING_FACILITY" type="xsd:string"/>
                <xsd:element name="PV1_19_1_VISIT_NUMBER" type="xsd:string"/>
                <xsd:element name="PV1_44_ADMIT_DATE_TIME" type="xsd:string"/>
                <xsd:element name="PV1_45_DISCHARGE_DATE_TIME" type="xsd:string"/>
                <xsd:element name="LOS" type="xsd:int"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

</xsd:schema>
```

Save the XSD.

In the new project right-click the Process Files node and choose New WSDL Document. Name this WSDL EMailNotifier_BPELM_EMailOut. Make this WSDL a Concrete WSDL, EMAIL Binding of type “Send Email (using SMTP)”.



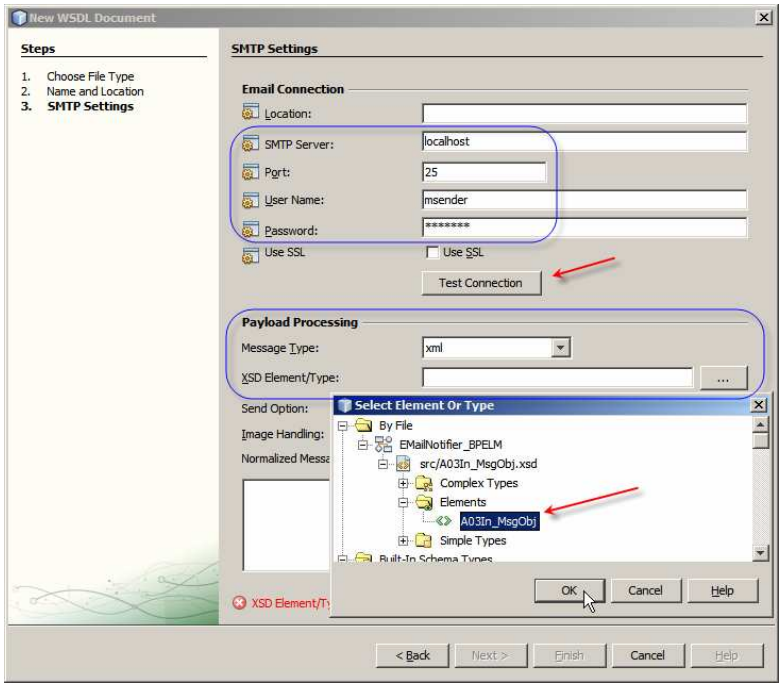
Click Next.

Configure outgoing email properties:

- SMTP Server: localhost (you would use a server of your own)
- Port: 25 (default for non-secure SMTP server)
- Username: msender (use your own)
- Password: ******* (use your own)
- Message Type: xml
- XSD Element/Type: A03In_MsgObj

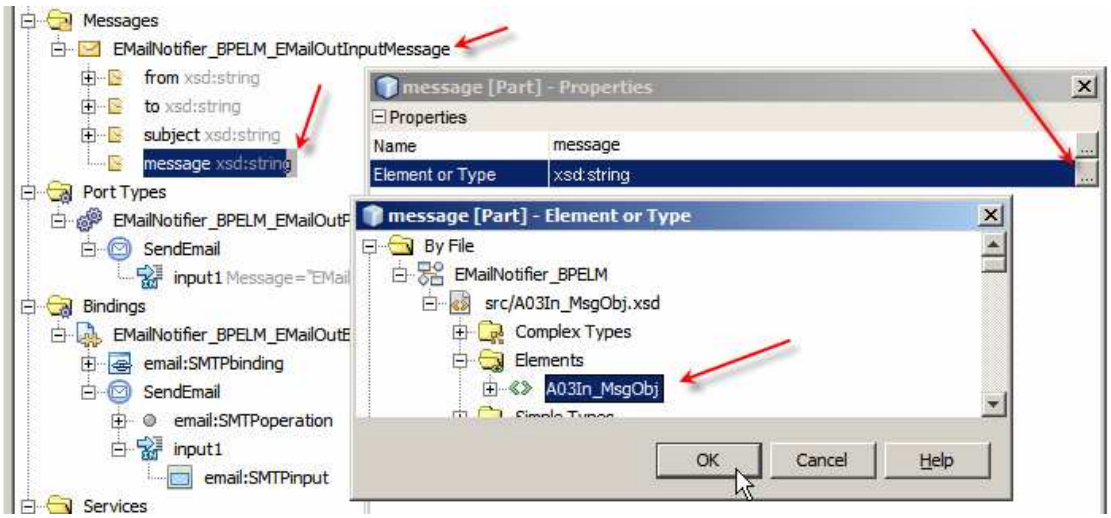
Click the Test button to verify that your connection settings are correct.

Leave other properties at defaults and click Finish to complete the wizard.



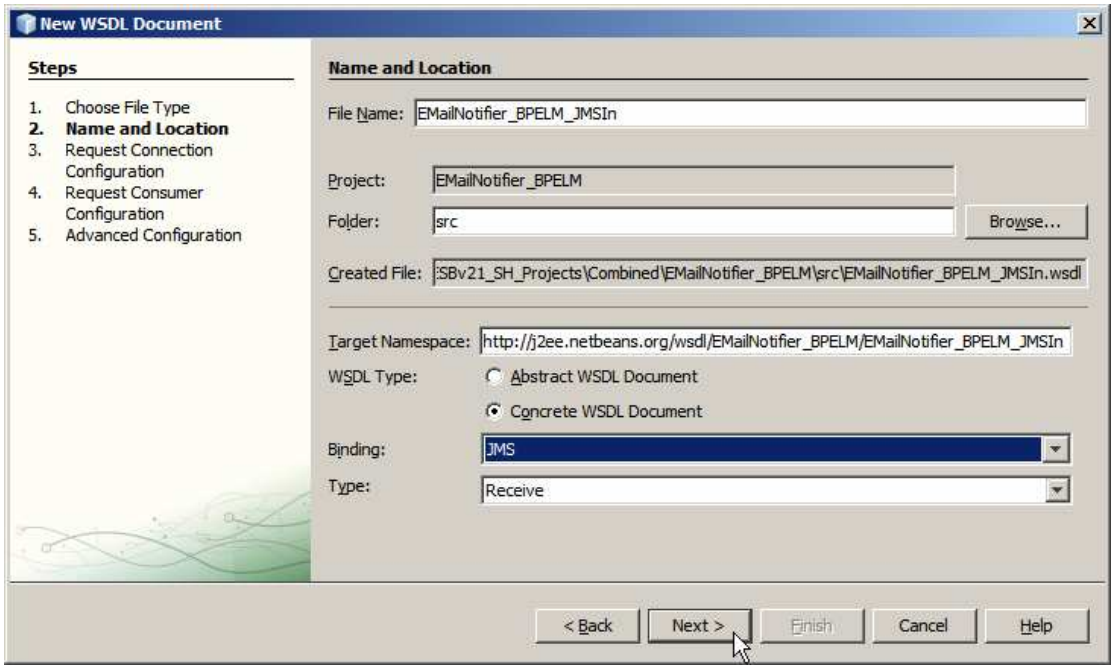
Alas, this does not correctly set the message type for the Email BC. Open the new WSDL in the editor.

Expand the Messages node. Right-click the “message” node and choose Properties. Change the Element or Type property to A03In_MsgObj. Save the WSDL.



Create a “New” -> “WSDL Document”, named EMailNotifier_BPELM_JMSIn.

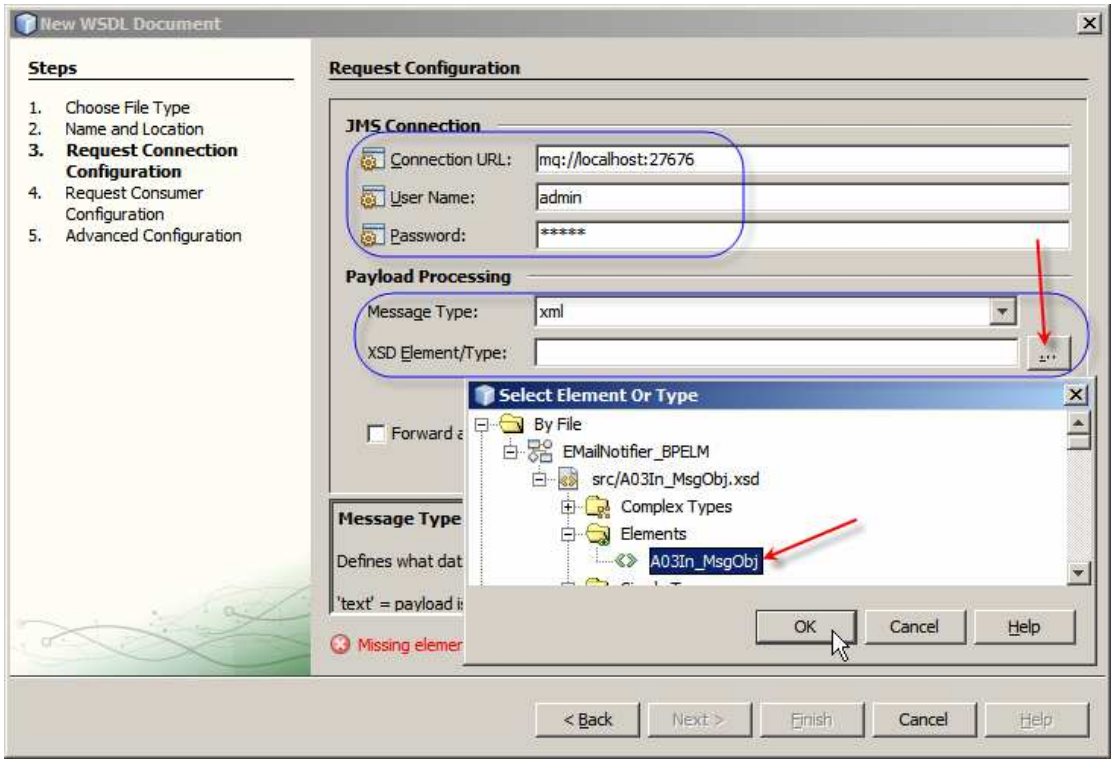
Make is a Concrete WSDL, JMS Binding, of type Receive.



Click Next.

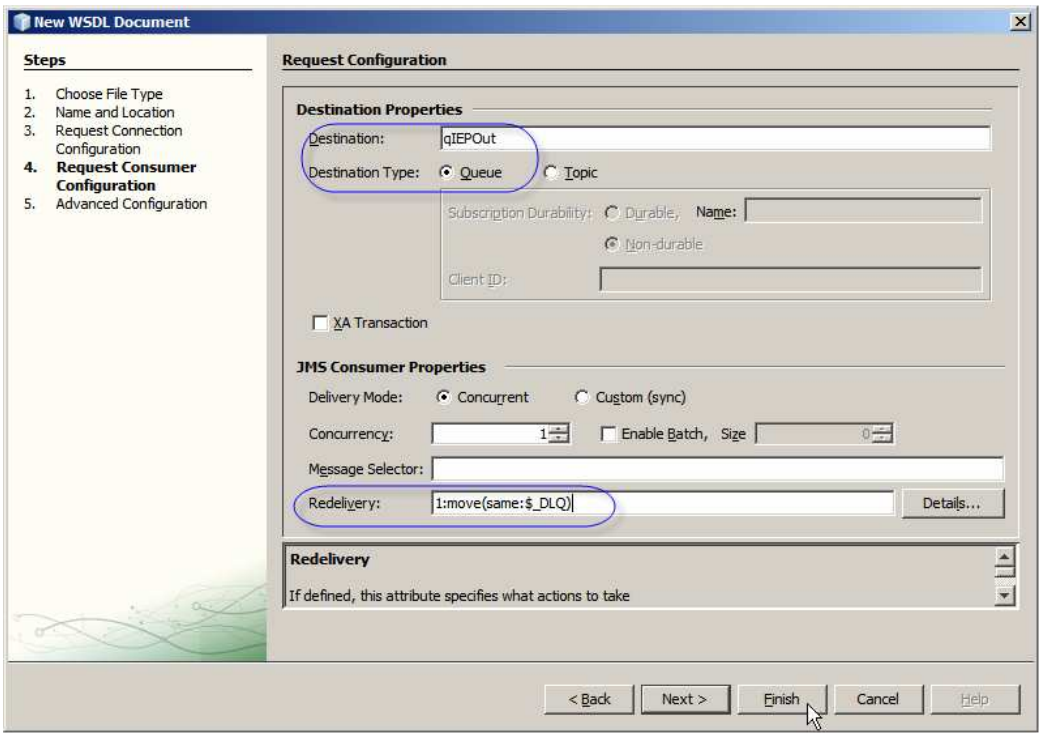
Configure properties:

- Connection URL: mq://localhost:27676 (change host and port for your environment)
- User Name: admin
- Password: admin
- Message Type: xml
- XSD Element/Type: A03In_MsgObj

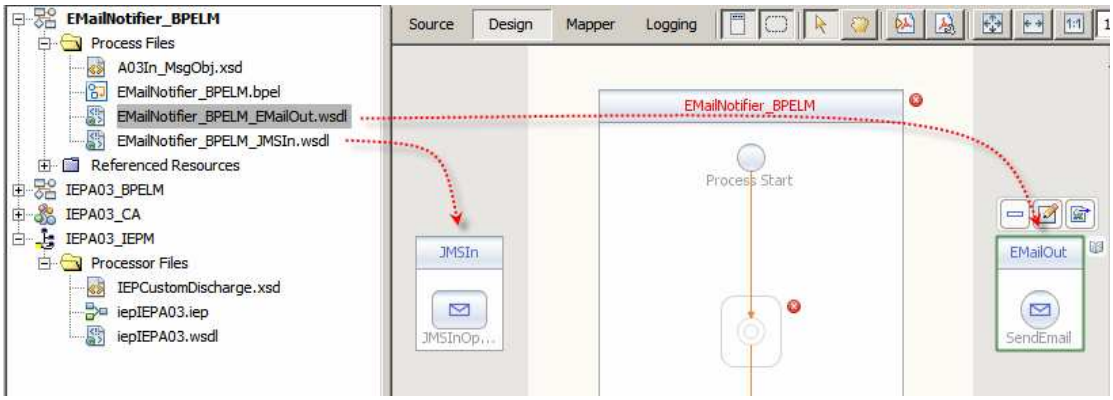


Click Next.

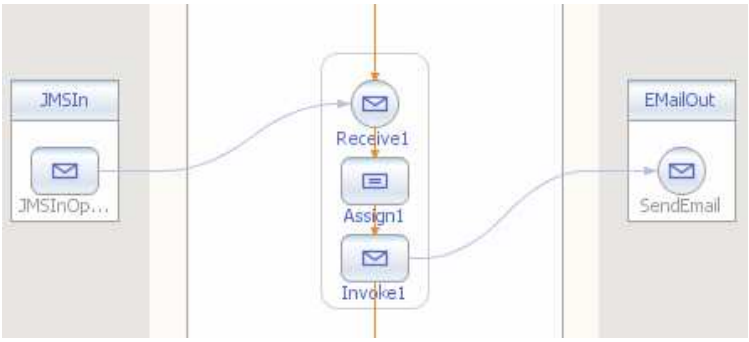
Configure Destination and Redeliver properties, and click Finish.



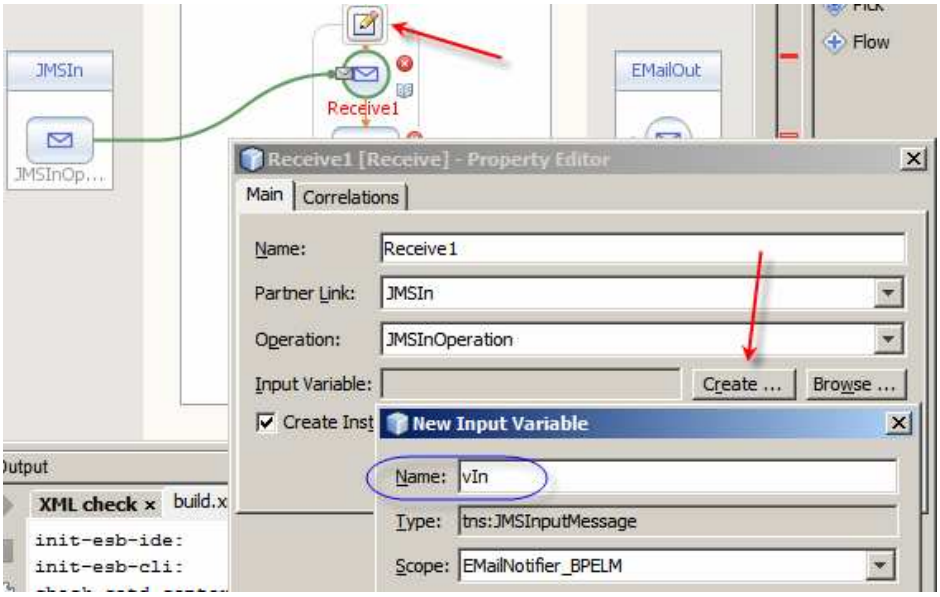
Open the BPEL process. Drag the EMailNotifier_BPELM_JMSIn onto the canvas and drop it onto the target marker in the left hand swim line. Name the partner link JMSIn. Drag the EMailNotifier_BPELM_EMailOut WSDL and drop it onto the target marker in the right hand swim line. Name the partner link EMailOut.



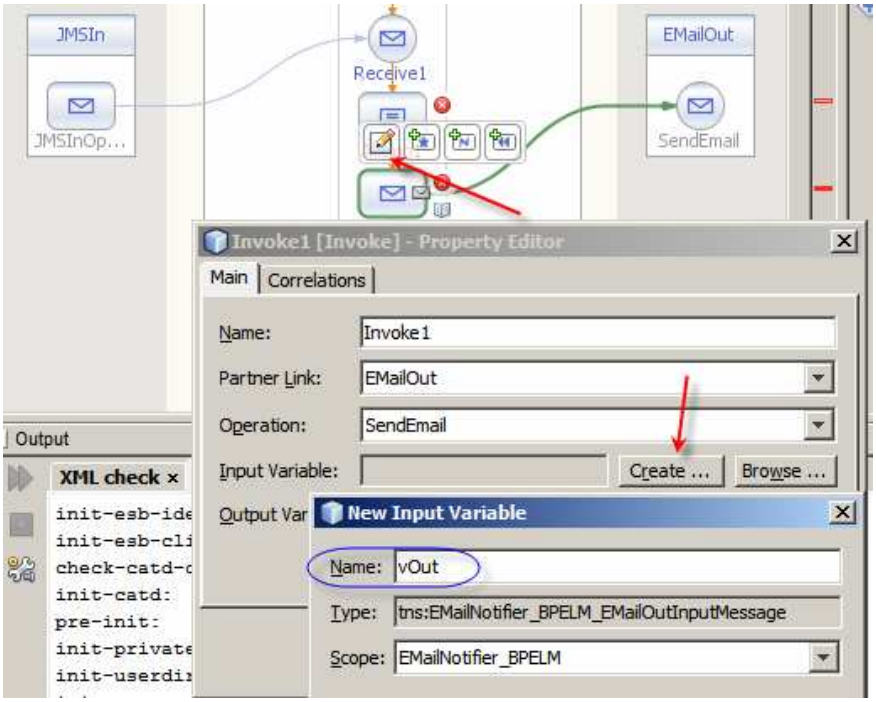
Drag Receive, Assign and Invoke activities onto the canvas and drop onto the target markers inside the process scope. Connect the Receive1 to JMSInOperation operation of the inbound partner and the Invoke1 to the SendEmail operation of the outbound partner link.



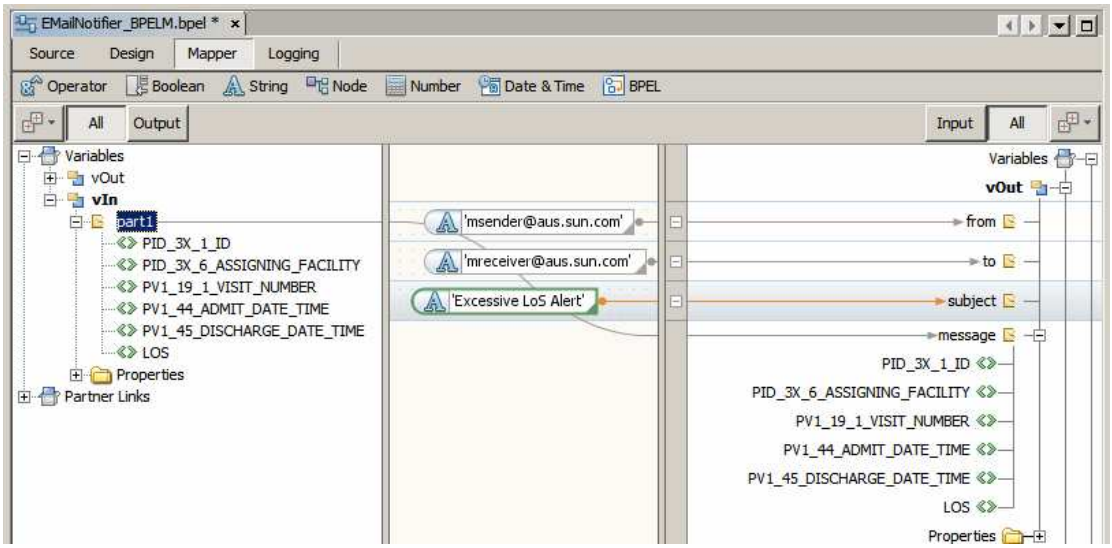
Add a variable to the Receive1 activity to hold the incoming message.



Add a variable to the Invoke1 activity to contain the outgoing message.

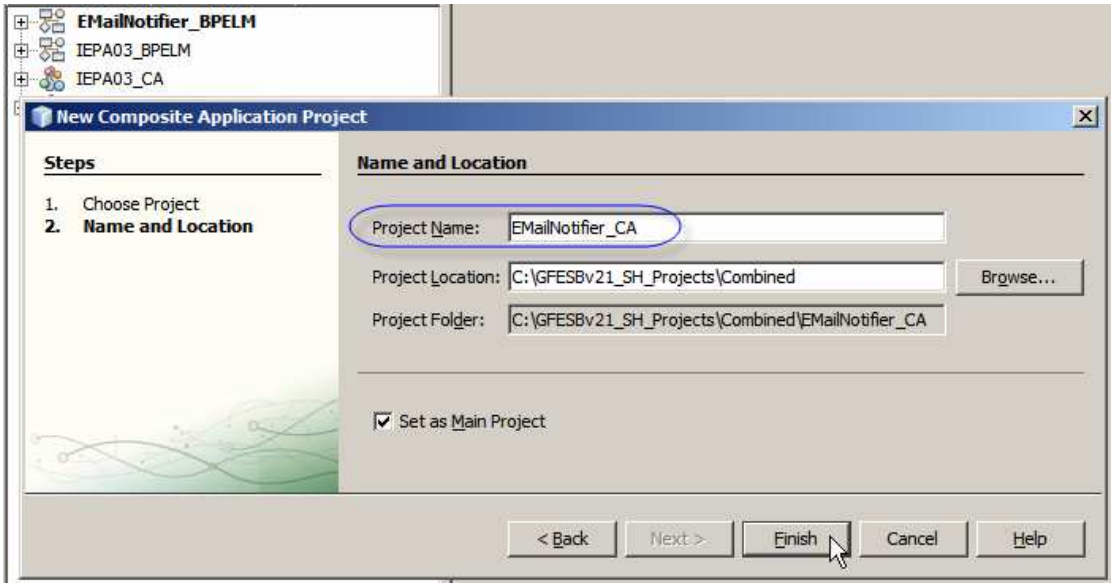


Double-click the Assign1 activity and configure mappings as shown, varying from email address and subject as you see fit.



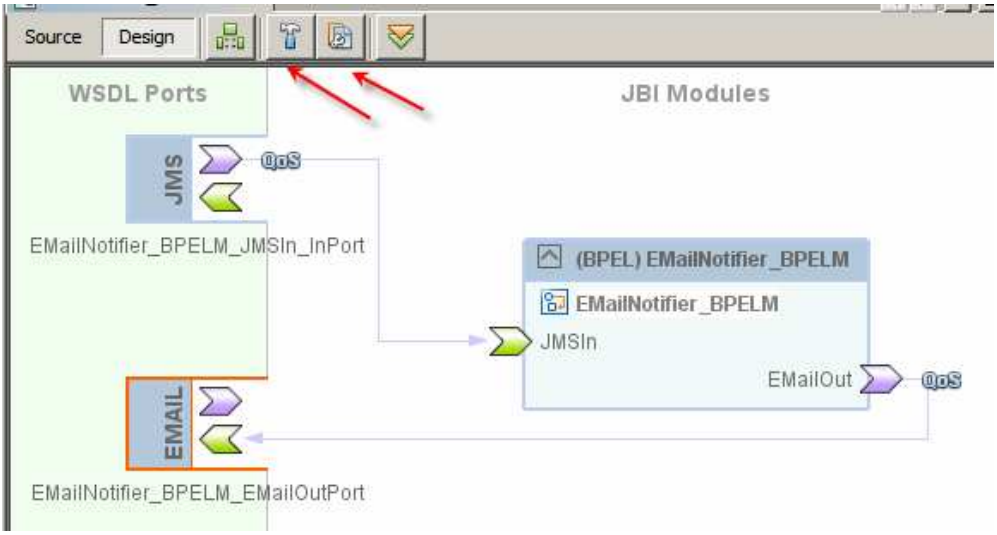
Build the project.

Create a “New Project” -> “SOA” -> “Composite Application”, named EMailNotifier_CA.

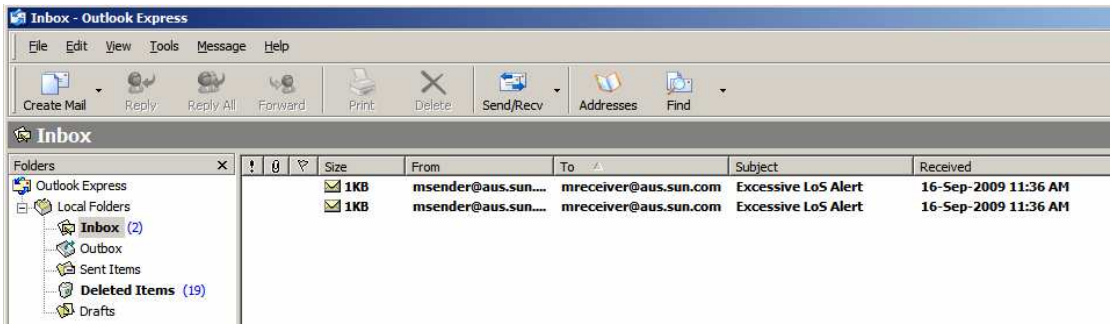


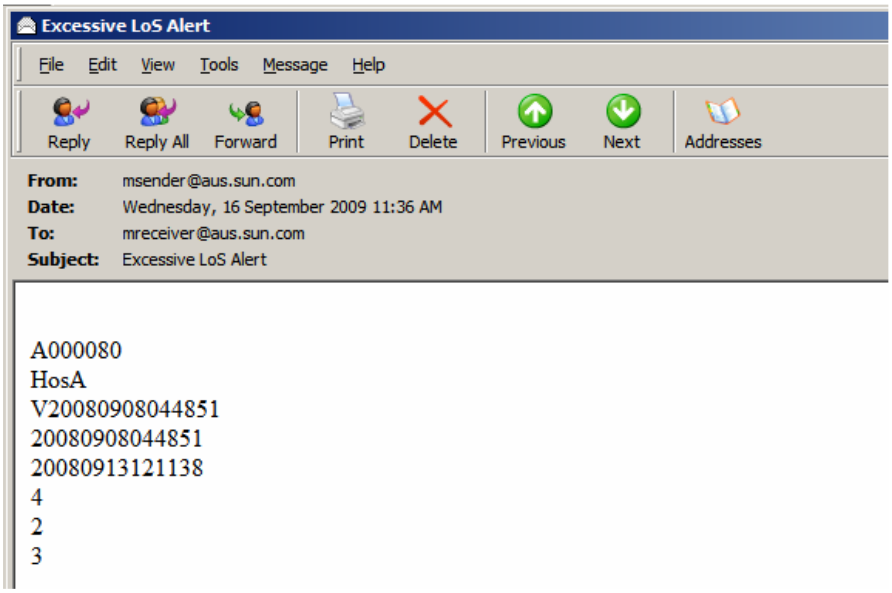
Drag the EMailNotifier_BPELM project onto the CASA canvas.

Build and Deploy the project.



Submit the 10-record file, IEP_output_10.dat, as before. Again, wait for the IEP output stream to release the batch of records. Use you electronic mail client to receive and review notification messages.





This message corresponds to the following A03In_MsgObj XML message:

```
<A03In_MsgObj
  xmlns:msgns="iepIEPA03_iep"
  xmlns:ns0="http://j2ee.netbeans.org/wsdl/EMailNotifier_BPELM/EMailNotifier_BPELM_EMailOut"
  xmlns="iepIEPA03_iep">
    <PID_3X_1_ID xmlns="">A000080</PID_3X_1_ID>
    <PID_3X_6_ASSIGNING_FACILITY xmlns="">HosA</PID_3X_6_ASSIGNING_FACILITY>
    <PV1_19_1_VISIT_NUMBER xmlns="">V20080908044851</PV1_19_1_VISIT_NUMBER>
    <PV1_44_ADMIT_DATE_TIME xmlns="">20080908044851</PV1_44_ADMIT_DATE_TIME>
    <PV1_45_DISCHARGE_DATE_TIME xmlns="">20080913121138</PV1_45_DISCHARGE_DATE_TIME>
    <LOS xmlns="">4</LOS>
    <AvgLoS xmlns="">2</AvgLoS>
    <AvgLoS1andaHalf xmlns="">3</AvgLoS1andaHalf>
</A03In_MsgObj>
```

The Email BC stripped XML markup and gave us just the date content.

Exercising the solution through the HL7 Processor

The HL7 Processor solution is discussed in "HL7 Processor Demonstration - GlassFish ESB v2.1", http://blogs.sun.com/javacapsfieldtech/entry/hl7_processor_demonstration_glassfish_esb. Build and deploy this solution. Submit to it the 50 record file, ADT_A0x_output50.dat, and observe the outcome produced by the IEP solution.

By sharing a common JMS Queue, qIEP, to which the HL7 Processor wrote custom discharge messages, and from which the IEP Processor read custom discharge messages, the three separate solutions were able to work together to accomplish a goal neither of the solutions would be able to accomplish by itself.

Summary

So, what have we seen and accomplished?

We constructed an Intelligent Event Processor solution to process discharge events, calculating an average of the last 10 seconds worth of events each time a new even was processed, comparing that event's length of stay to the average at the time and isolating these events whose length of stay was more the 1 ½ times the average at the time. These events were written to the output stream, for some other solution to use for notification and alerting, for example.

We modified this basic solution to receive a stream of events over JMS and to emit the notification events to a JMS destination.

We then created a notification processor solution which used these events to send notifications by electronic mail.

Finally, we submitted a set of HL7 v2 delimited messages to the HL7 Processor solution, which broke them up into admissions and discharges, converted discharge messages to the custom discharge message format used by the IEP solution which we developed here, and delivered them to this IEP solution for processing.