GlassFish ESB v2.2 Field Notes

Ephemeral, JVM-global, POJO-based Sequence Number Generator for BPEL

Michael.Czapski@sun.com

January 2010, Release 1.0.0.0

Table of Contents

Introduction

When working on the HA solutions discussed in my blog[1] I realized that it will be difficult to work out whether messages are delivered in order, as was required, and whether any are missing. I got over the issue by ensuring that my test data was prepared in such a way that messages in each test file had increasing, contiguous sequence numbers embedded in the message. For HL7 v2, which is the messaging standard with which I dealt, I used MSH-10, Message Control ID field. I wrote processed messages and acknowledgements to files whose names embedded MSH-10 Message Control Id, with the sequence number, so breaks in sequence and out of order messages could be readily detected.

With multiple message files containing between 1 and 50,000 messages, adding a sequence number to each message by hand was clearly out of the question.

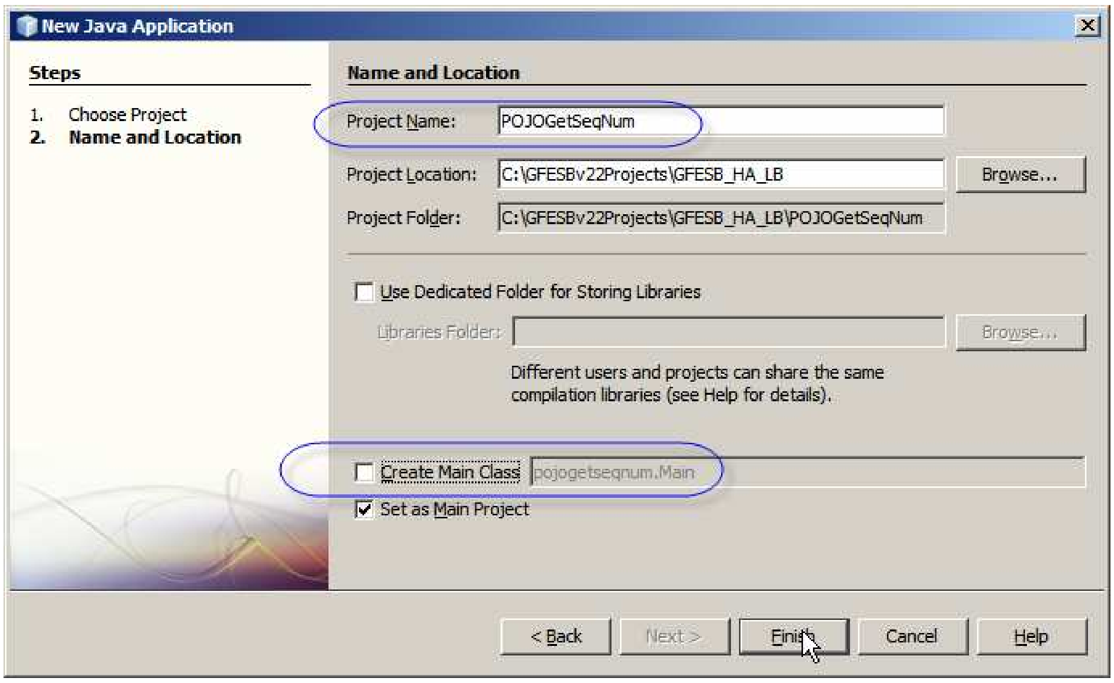
I put the GlassFish ESB to use. I constructed a file-to-file BPEL module project to read each test file and to prepend a sequence number to each message's MSH-10 field. The only snag was how to get a sequence number that would start at 0 and increase by 1 for each message, such that each BPEL process instance would get the next sequence, and that messages would be written to the output file in order.

This note discusses how I went about accomplishing the task.

JVM-Global, Ephemeral Sequence Generator

After considering a couple of possibilities I resolved to use a Java `static int` to hold the sequence number, knowing that it will be good until the project is re-deployed or until the Application Server is restarted, which was good enough for my purpose. To make the Java code accessible to BPEL I decided to create a POJO, which to invoke from BPEL.

Let's create a "New Project" -> "Java" -> "Java Application", POJOGetSeqNum, making sure to uncheck the "create main class" option.


New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: POJOGetSeqNum

Project Location: C:\GFESBv22Projects\GFESB_HA_LB

Project Folder: C:\GFESBv22Projects\GFESB_HA_LB\POJOGetSeqNum

☐ Use Dedicated Folder for Storing Libraries

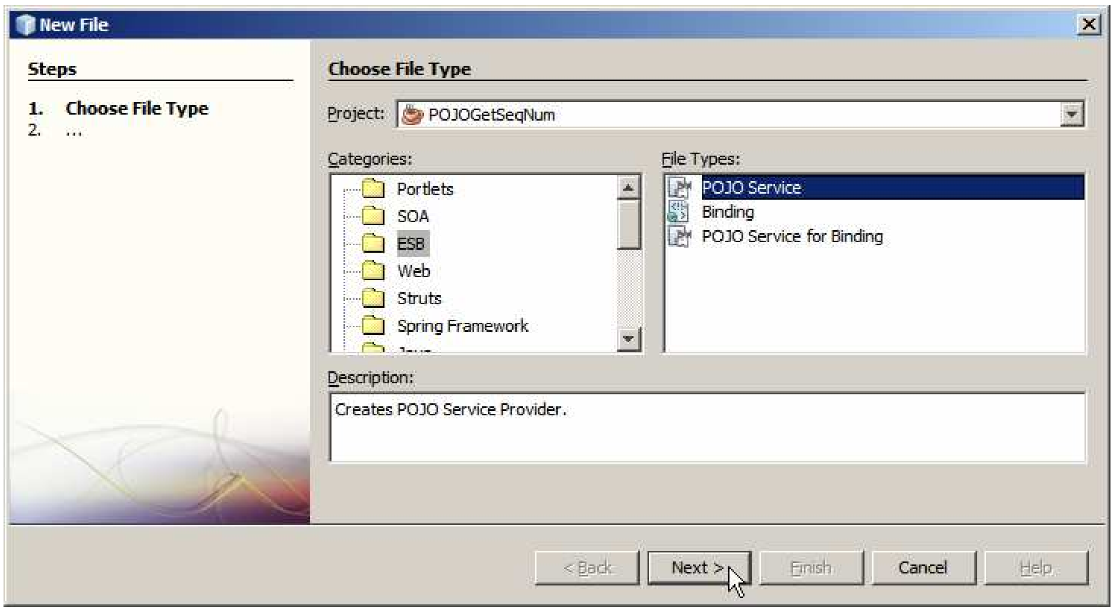
Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☐ Create Main Class pojogetseqnum.Main

☑ Set as Main Project

Right-click project name, choose “New” → “Other …”. Choose “ESB” → “POJO Service”.


New File

Steps

1. **Choose File Type**
2. ...

Choose File Type

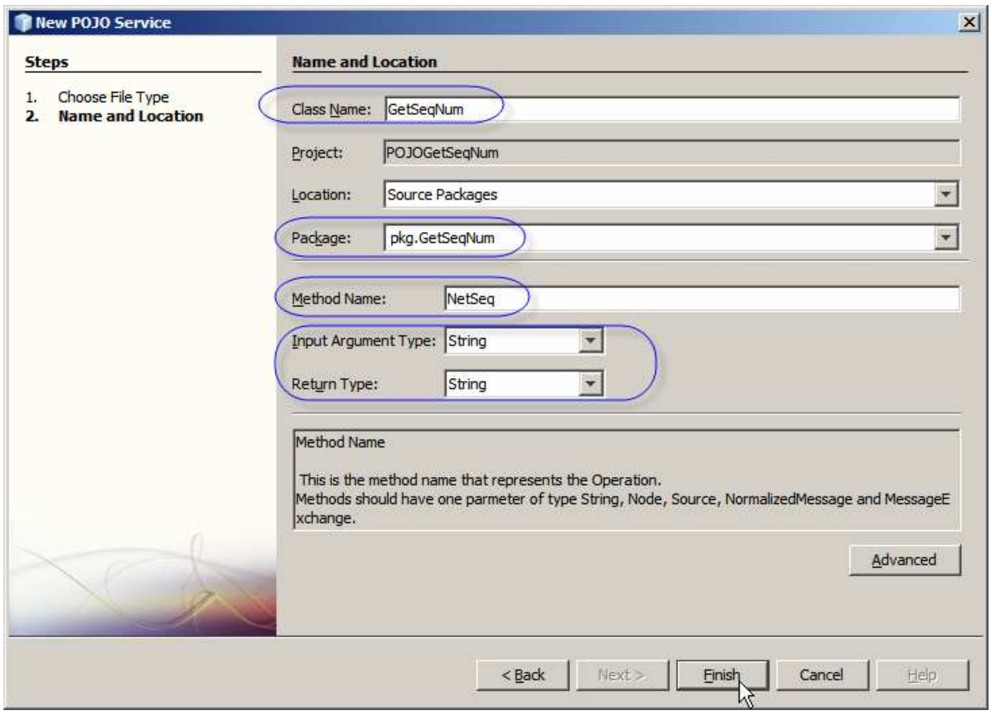
Project: POJOGetSeqNum

Categories: Portlets, SOA, ESB, Web, Struts, Spring Framework, Java

File Types: POJO Service, Binding, POJO Service for Binding

Description: Creates POJO Service Provider.

Specify Class Name: GetSeqNum, package name: pkg.GetSeqNum, method name: NextSeq, and accept input and output parameters as strings.



Name and Location

Class Name: GetSeqNum

Project: POJOGetSeqNum

Location: Source Packages

Package: pkg.GetSeqNum

Method Name: NetSeq

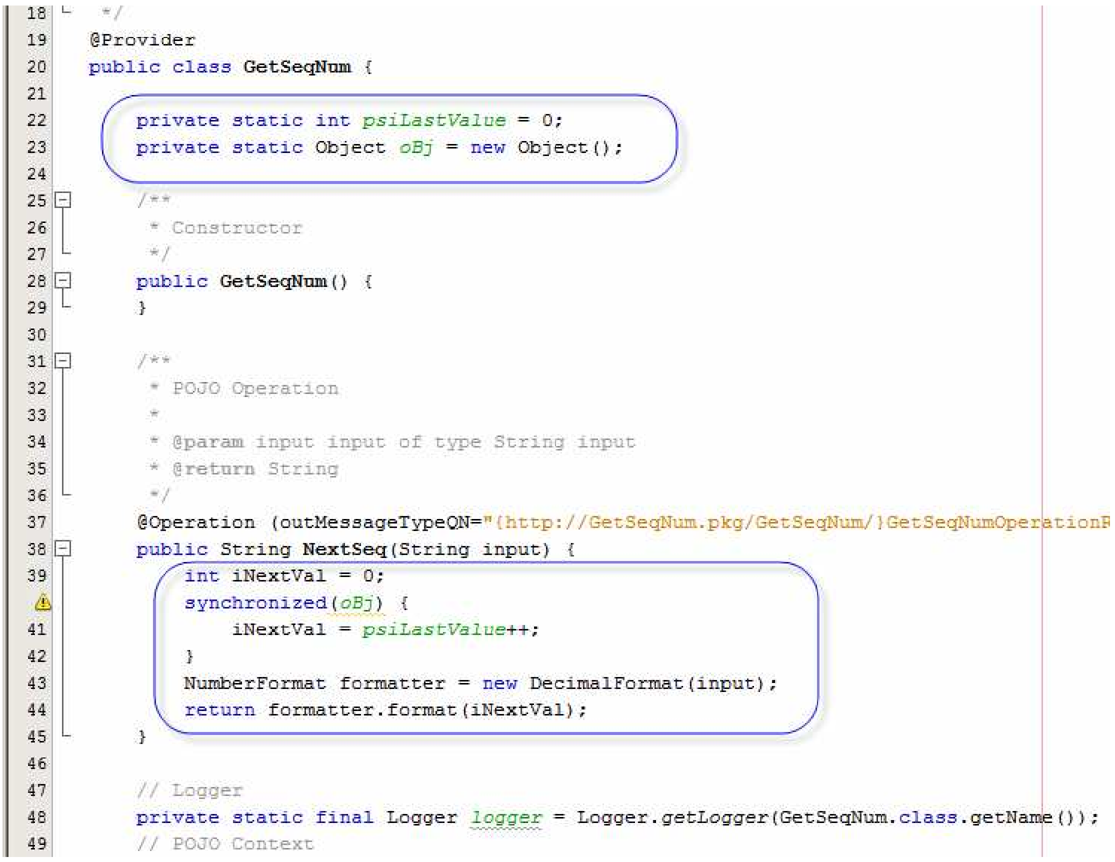
Input Argument Type: String

Return Type: String

Method Name

This is the method name that represents the Operation.
Methods should have one parmeter of type String, Node, Source, NormalizedMessage and MessageExchange.

When the skeleton POJO code is shown in the editor, add the following source code as indicated:



```
 */
@Provider
public class GetSeqNum {

    private static int psiLastValue = 0;
    private static Object oBj = new Object();

    /**
     * Constructor
     */
    public GetSeqNum() {
    }

    /**
     * POJO Operation
     *
     * @param input input of type String input
     * @return String
     */
    @Operation (outMessageTypeQN="{http://GetSeqNum.pkg/GetSeqNum/}GetSeqNumOperationR
    public String NextSeq(String input) {
        int iNextVal = 0;
        synchronized(oBj) {
            iNextVal = psiLastValue++;
        }
        NumberFormat formatter = new DecimalFormat(input);
        return formatter.format(iNextVal);
    }

    // Logger
    private static final Logger logger = Logger.getLogger(GetSeqNum.class.getName());
    // POJO Context
```

First we declare private *static int* to hold the most recent sequence number, and a synchronization object, since the formatter may not be thread-safe.

Inside the NetSeq method body we initialize the return variable, assign the most recent sequence value to it with post-increment, inside a synchronization block, create a formatter to format the number as numeric string, format the number and return it to the caller.

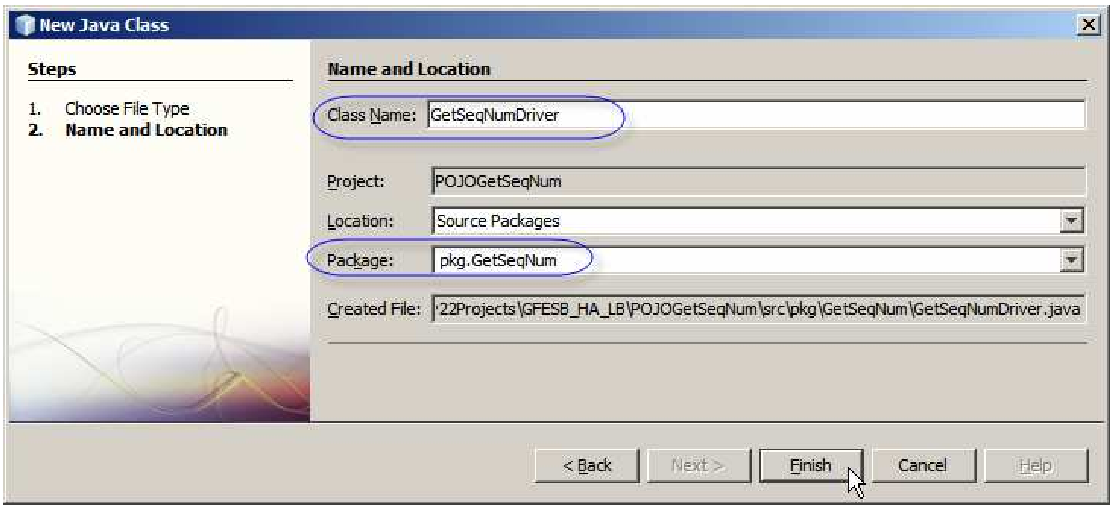
Once the code is added, right click inside the source window and choose “Fix Imports”, to resolve missing import statements.

If I was interested in the sequence number as an *int* I could have skipped the formatting bit and returned an *int*. To keep my messages in order by sequence number, I prefer to have a zero-filled, right-justified number strings, like 000001.

Compile the file. It should compile unless something got messed up in transcription.

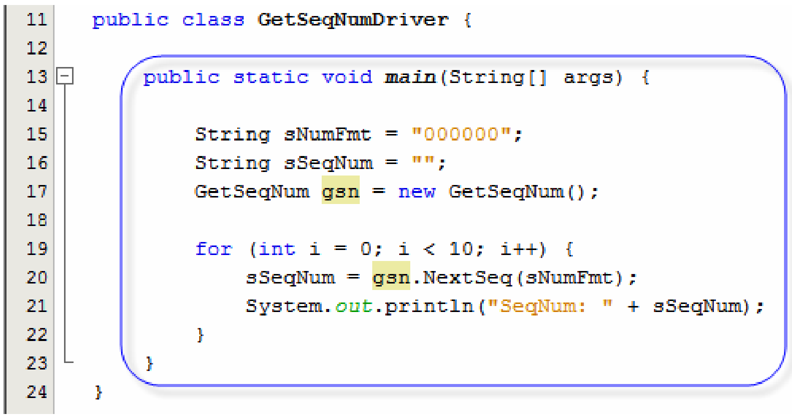
To test the code let’s add a Main class to the project and use it to invoke the NextSeq method in our class.

Right-click the name of the project and choose “New” → “Java Class”. Name it GetNextSeqDriver and make sure to include it in the same package as GetNetSeq.



Class Name: GetSeqNumDriver
Project: POJOGetSeqNum
Location: Source Packages
Package: pkg.GetSeqNum
Created File: 22Projects\GFESB_HA_LB\POJOGetSeqNum\src\pkg\GetSeqNum\GetSeqNumDriver.java

Insert the following code as shown.

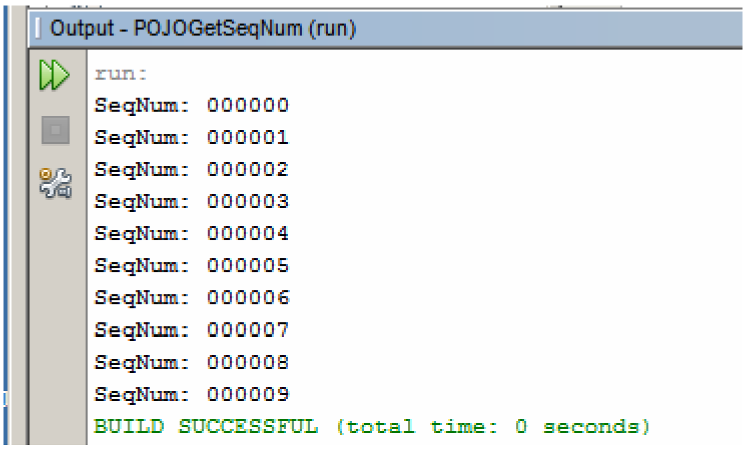

```
public class GetSeqNumDriver {

    public static void main(String[] args) {

        String sNumFmt = "000000";
        String sSeqNum = "";
        GetSeqNum gsn = new GetSeqNum();

        for (int i = 0; i < 10; i++) {
            sSeqNum = gsn.NextSeq(sNumFmt);
            System.out.println("SeqNum: " + sSeqNum);
        }
    }
}
```

Right-click the name of this calss, GetSeqNumDriver, choose “Run File” and observe the outcome in the output window.



```
run:
SeqNum: 000000
SeqNum: 000001
SeqNum: 000002
SeqNum: 000003
SeqNum: 000004
SeqNum: 000005
SeqNum: 000006
SeqNum: 000007
SeqNum: 000008
SeqNum: 000009
BUILD SUCCESSFUL (total time: 0 seconds)
```

Right-click the project name and choose “Build”.

The utility class is ready.

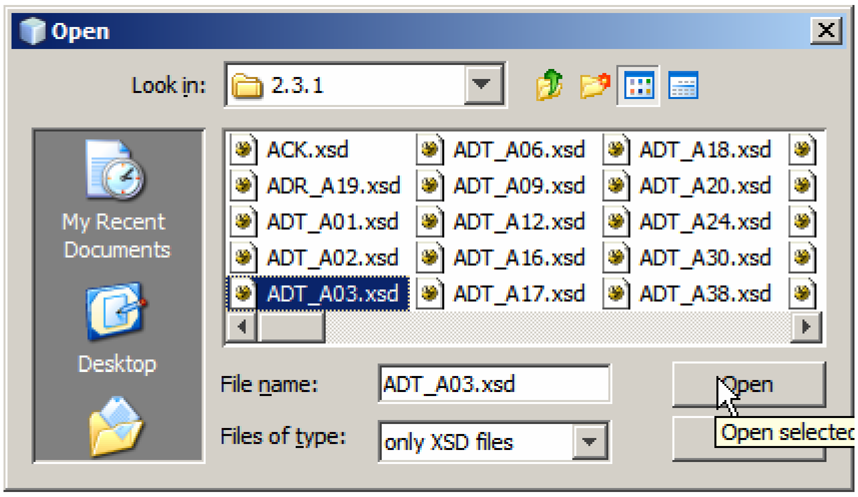
HL7 Message ID Encricher

With the sequence number generator ready we are now in a position to process HL7 v2 messages by prepending increasing sequence number to the MSH-10 Message Control ID, so we can use these messages in HA testing.

Obtain HL7 v2 XML Schema documents, from which we will need ADT A03 and related schemas, from http://wiki.open-esb.java.net/attach/HL7/hl7v2xsd.zip. Unzip it to a convenient location.

Create a “New Project” → “SOA” → “BPEL Module”, named HL7Sequencer.

Create a sub-folder, HL7v231, in the “Process Files” folder. Right click the name of the subfolder and choose “New” → “External XML Schema Document(s)”. Locate the ADT_A03.xsd in the hl7 v2.3.1 folder hierarchy hl7v2xsd/2.3.1, and choose it.

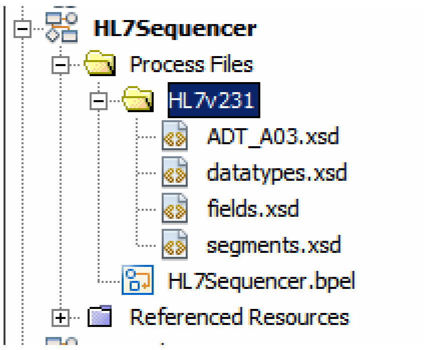


File name: ADT_A03.xsd

Files of type: only XSD files

Click Finish.

Supplementary files, included in ADT_A03.xsd, were added as well. The project will now look like this:

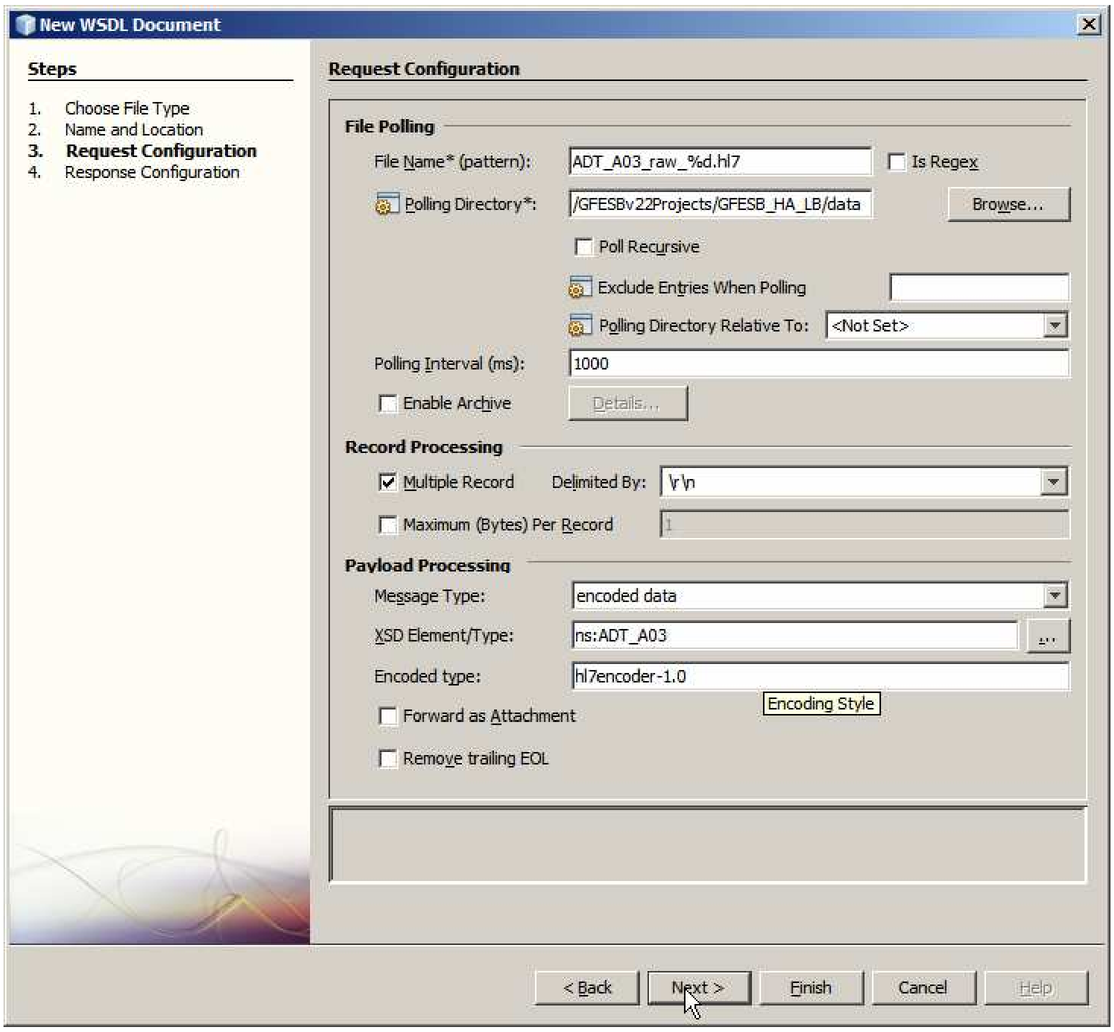


We expect to read one or more HL7 version 2.3.1 delimited ADT A03 messages, enrich the MSH-10 field, and write out HL7 v 2.3.1 delimited ADT A03 messages. This requires a File BC to read and write records. We could use one File BC configuration for reading and one for writing, but we can also use a single File BC configuration to both read records from a file and to write records to a different file in the same directory.

Let's create "New" → "WSDL Document", named HL7Sequencer_FileInOut:

WSDL Type: Concrete
Binding: FILE
Type: poll and write back reply

Request Configuration
File Polling:
    File Name: ADT_A03_raw_%d.hl7
    Polling Directory: /GFESBv22Projects/GFESB_HA_LB/data (or whatever directory you need to use)
Record processing:
    Multiple Records: true
    Delimiter: \r\n
Payload Processing:
    Message Type: encoded data
    XSD Element/Type: ADT_A03
    Encoded type: hl7encoder
    Remove Trailing EOL: true



File Polling

File Name* (pattern): ADT_A03_raw_%d.hl7 ☐ Is Regex
Polling Directory*: /GFESBv22Projects/GFESB_HA_LB/data
☐ Poll Recursive
Exclude Entries When Polling:
Polling Directory Relative To: <Not Set>
Polling Interval (ms): 1000
☐ Enable Archive

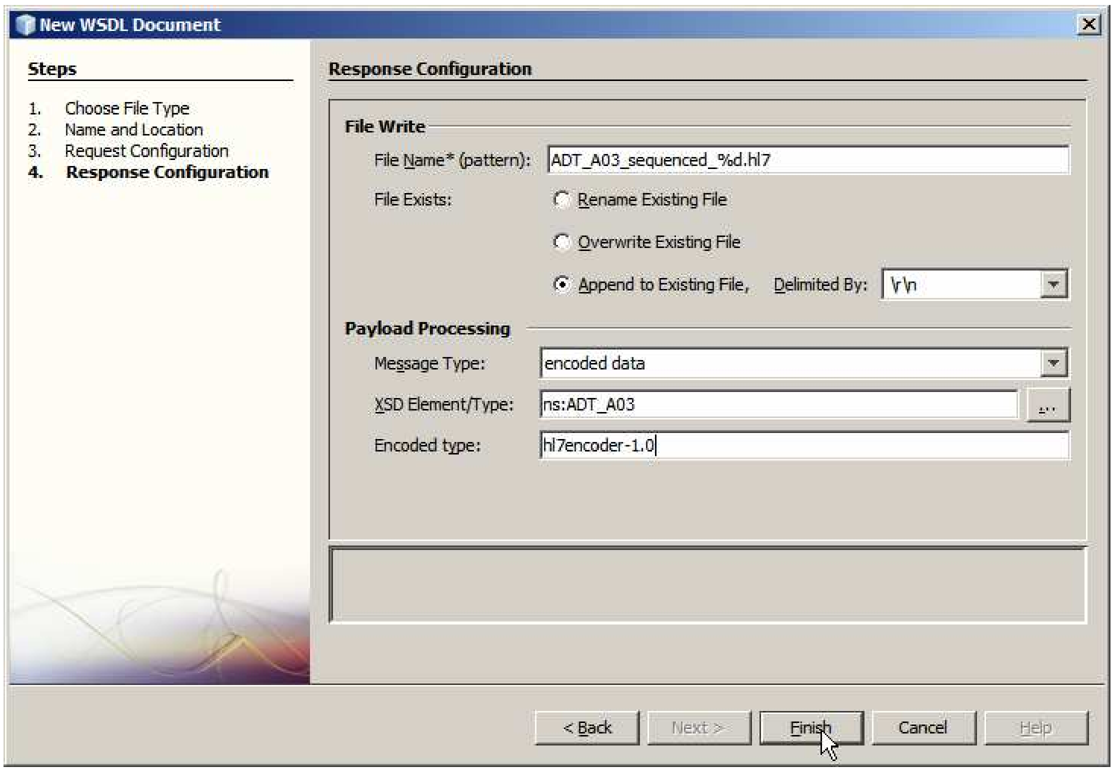
Record Processing

☑ Multiple Record Delimited By: \r\n
☐ Maximum (Bytes) Per Record 1

Payload Processing

Message Type: encoded data
XSD Element/Type: ns:ADT_A03
Encoded type: hl7encoder-1.0
☐ Forward as Attachment
☐ Remove trailing EOL

Response Configuration
File Write:
    File Name: ADT_A03_raw_%d.hl7
    File Exists: Append to existing file
    Delimited By: \r\n
Payload Processing:
    Message Type: encoded data
    XSD Element/Type: ADT_A03
    Encoded type: hl7encoder
    Remove Trailing EOL: true



New WSDL Document

Steps
1. Choose File Type
2. Name and Location
3. Request Configuration
4. **Response Configuration**

Response Configuration

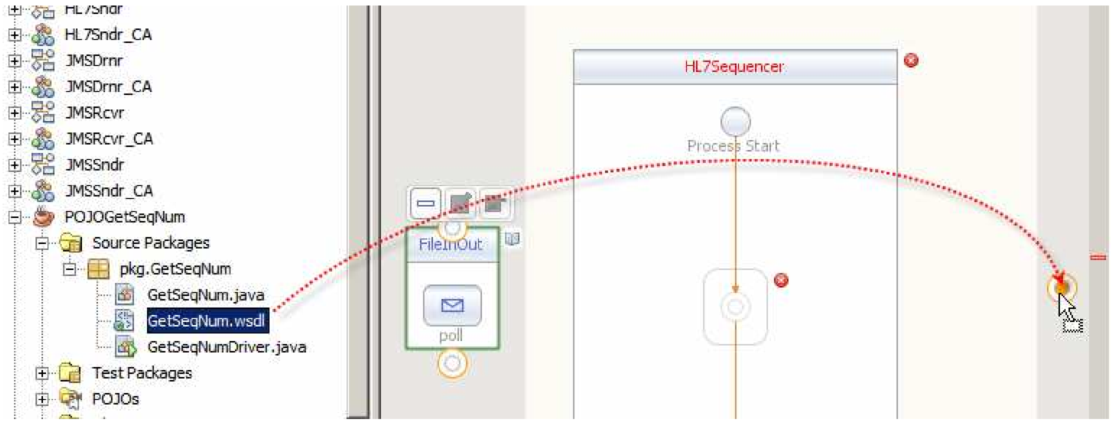
File Write
File Name* (pattern): ADT_A03_sequenced_%d.hl7
File Exists: ( ) Rename Existing File ( ) Overwrite Existing File [✓ Append to Existing File,] Delimited By: \r\n

Payload Processing
Message Type: encoded data
XSD Element/Type: ns:ADT_A03
Encoded type: hl7encoder-1.0

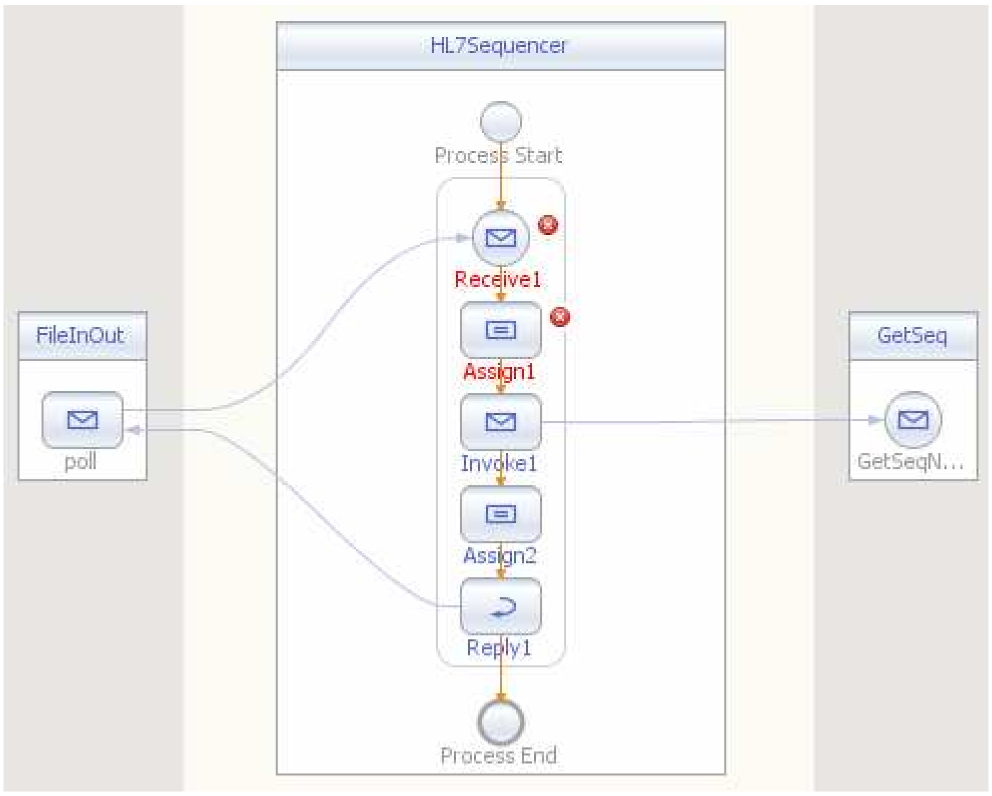
< Back | Next > | Finish | Cancel | Help

Open the BPEL Process and drag the WSDL onto the right-hand swim line. Name partner link FielInOut.

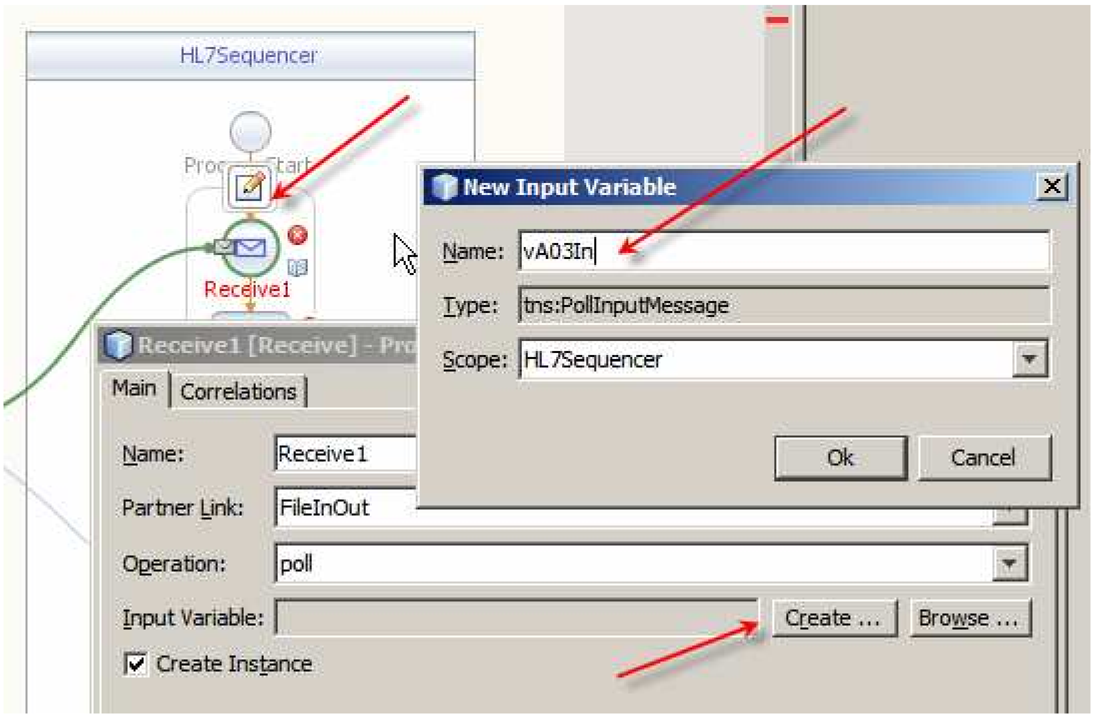
Expand the project POJOGetSeqNum, through the Source Packages → pkg.GetSeqNum node. Drag the GetSeqNum.wsdl onto the right hand swim line and name the partner link GetSeq.



Drag the Receive, Assign, Invoke, Assign and Reply activities onto the canvas and connect as shown.



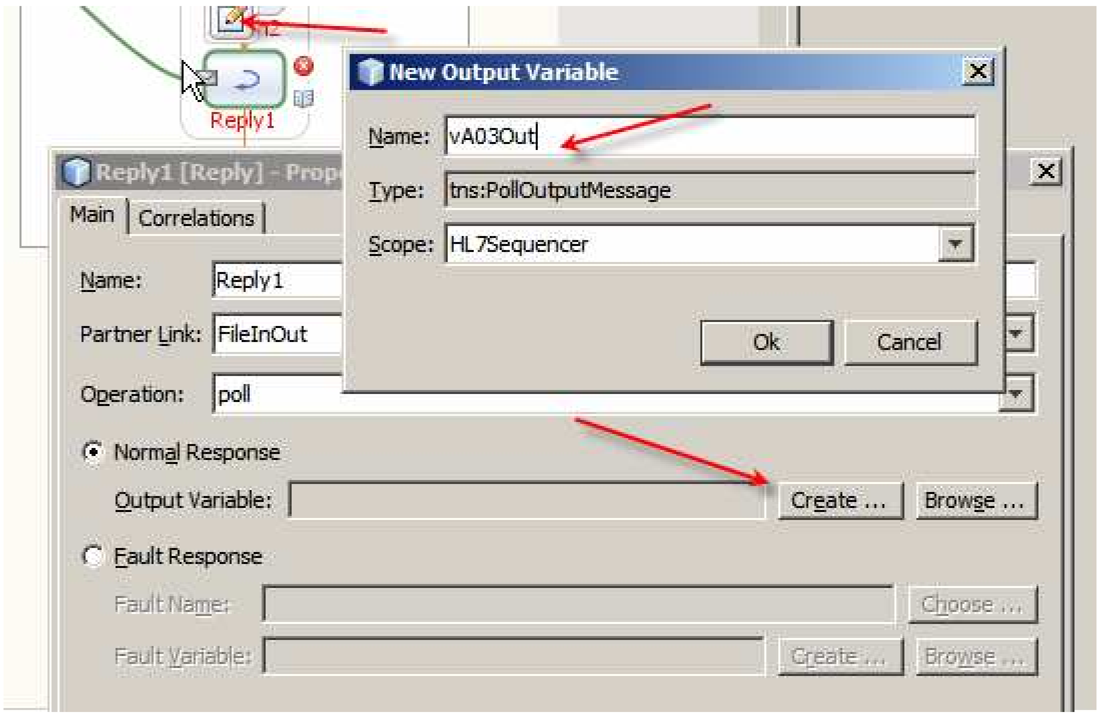
Edit Receive1 activity and create an Input Variable: vA03In.



Name: vA03In
Type: tns:PollInputMessage
Scope: HL7Sequencer

Name: Receive1
Partner Link: FileInOut
Operation: poll
Input Variable:
☑ Create Instance

Edit Reply1 activity and create Normal Response: Output Variable: vA03Out.


New Output Variable

Name: vA03Out

Type: tns:PollOutputMessage

Scope: HL7Sequencer

Reply1 [Reply] - Prop

Name: Reply1

Partner Link: FileInOut

Operation: poll

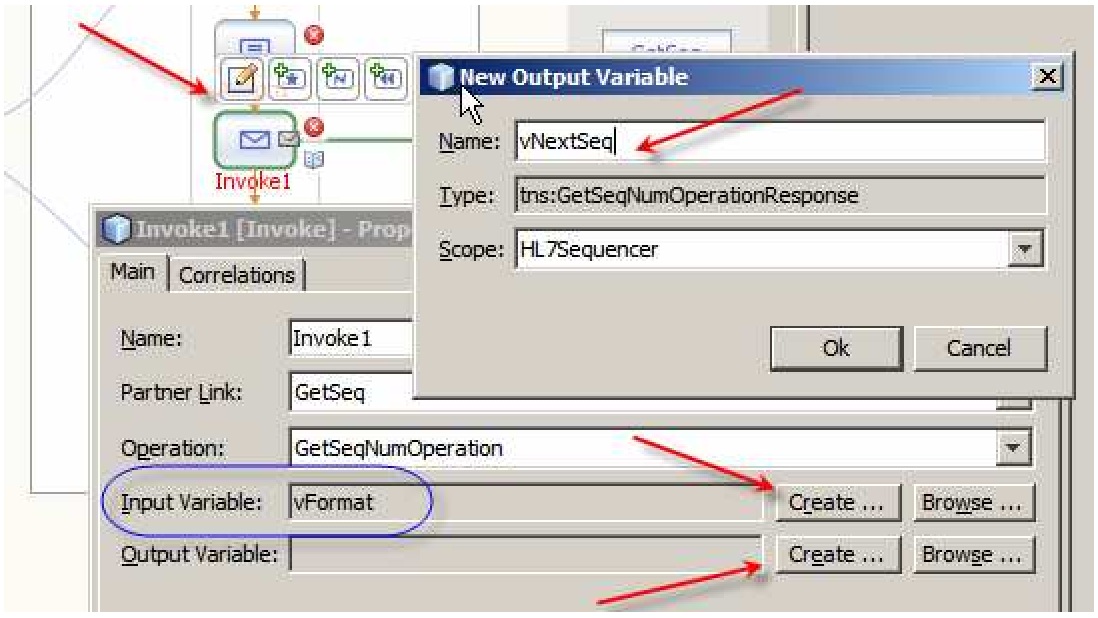
(● Normal Response) Output Variable:

( ) Fault Response

Fault Name:

Fault Variable:

Edit Invoke1 activity and create two variables, Input Variable: vFormat and Output Variable: vNextSeq.


New Output Variable

Name: vNextSeq

Type: tns:GetSeqNumOperationResponse

Scope: HL7Sequencer

Invoke1 [Invoke] - Prop

Name: Invoke1

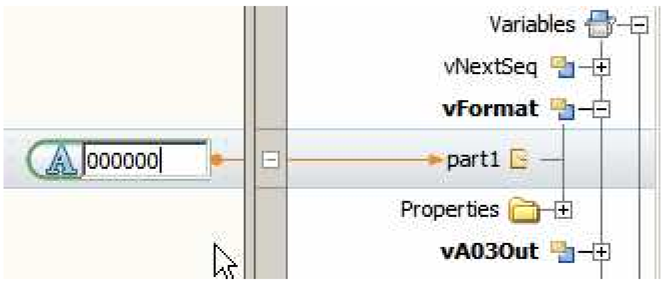
Partner Link: GetSeq

Operation: GetSeqNumOperation

Input Variable: vFormat

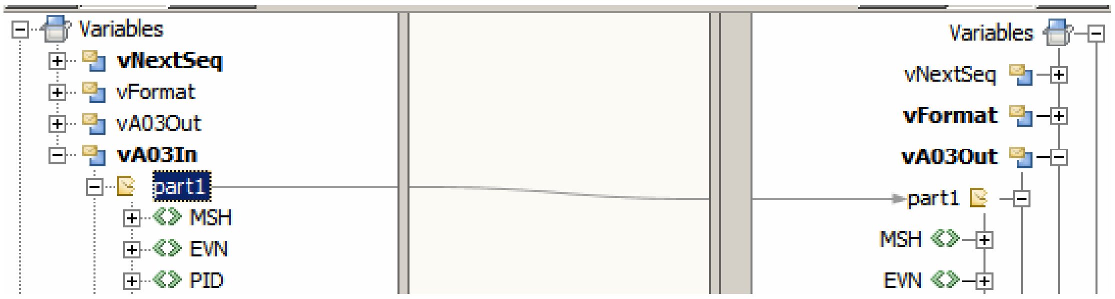
Output Variable:

Select Assign1 activity, switch to Mapper and assign string literal “000000” to variable vFormat.

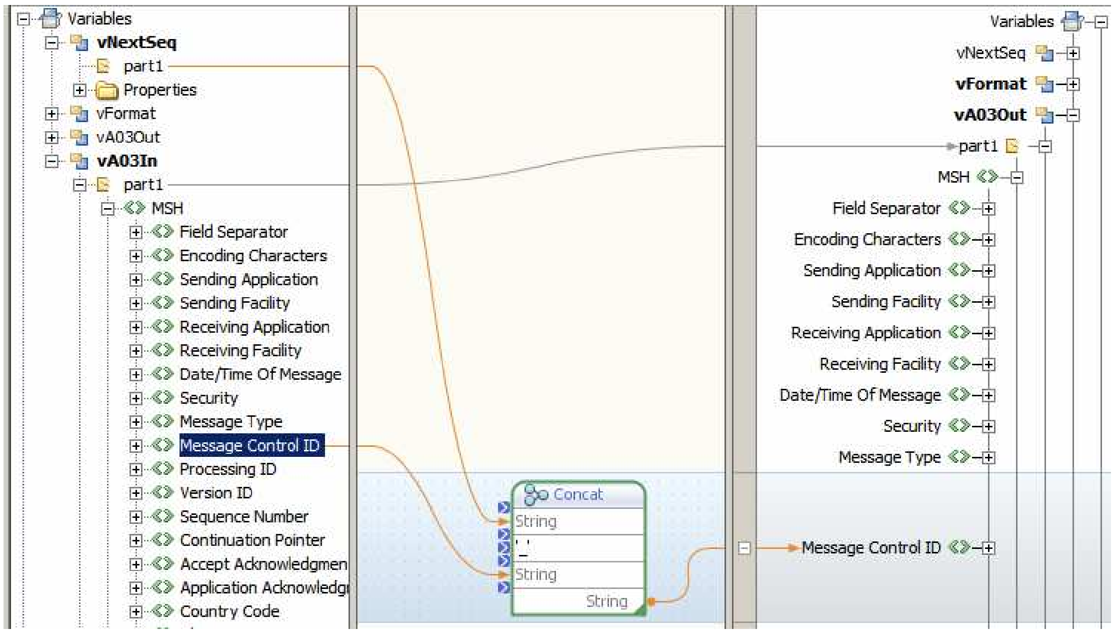


Switch to Design view, select Assign2 activity, switch to Mapper view and map, in stages, as discusses below.

Map vA03In→part1 to vA03Out→part1



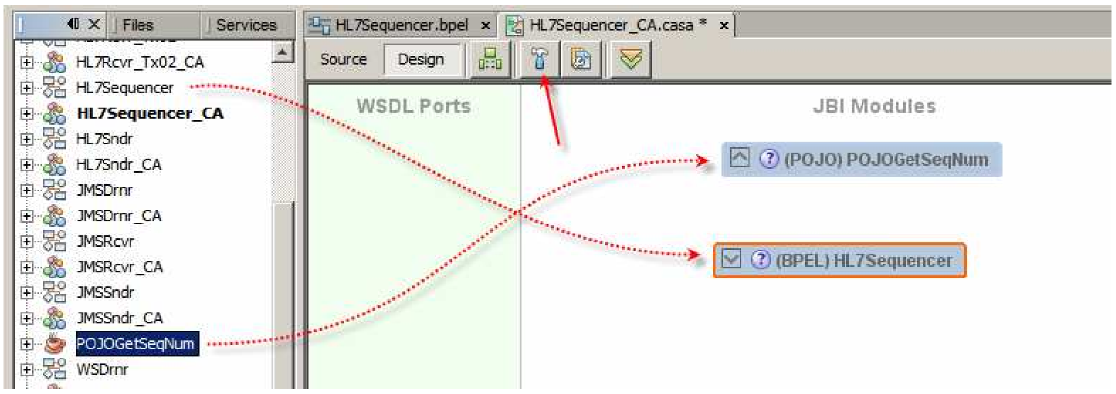
Map concatenation of vNextSeq→part1, literal "_" and vA03In→part1→MSH→Message Control ID to vA03Out→part1→MSH→Message Control ID.



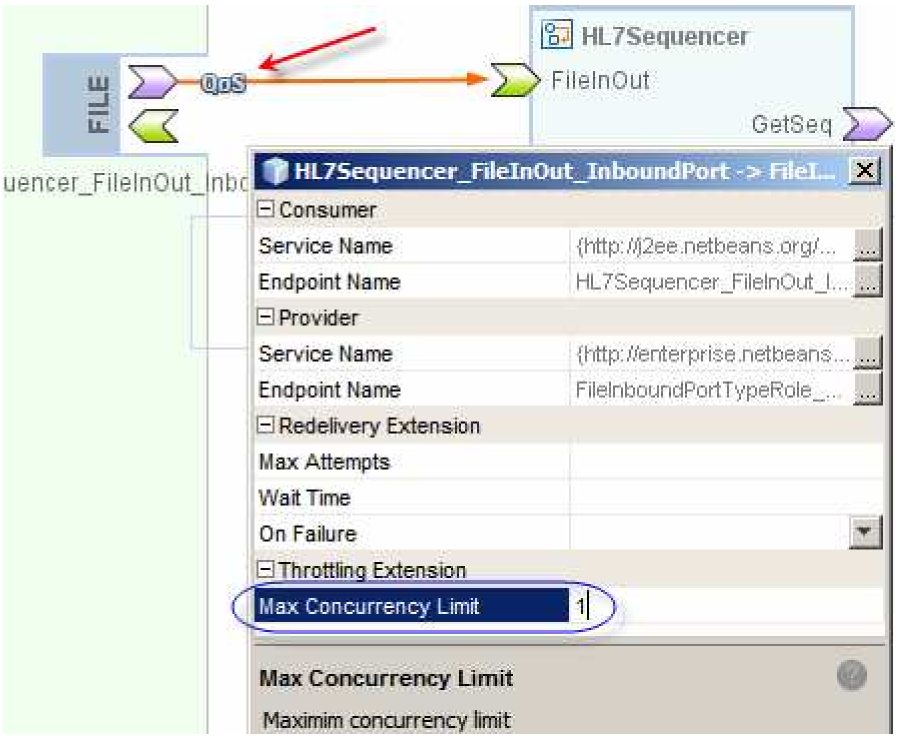
Save and Build the project.

Create "New Project"→"SOA"→"Composite Application", named HL7Sequencer_CA.

Drag HL7Sequencer and POJOGetSeqNum projects onto the CASA canvas and click Build.



Righ-click the QoS icon on the link form the File BC to the BPEL process and set Max Concurrency Limit to 1, to serialize message processing. Having gone to all this trouble to assign sequential numbers we need to make sure the records are indeed written out in order.



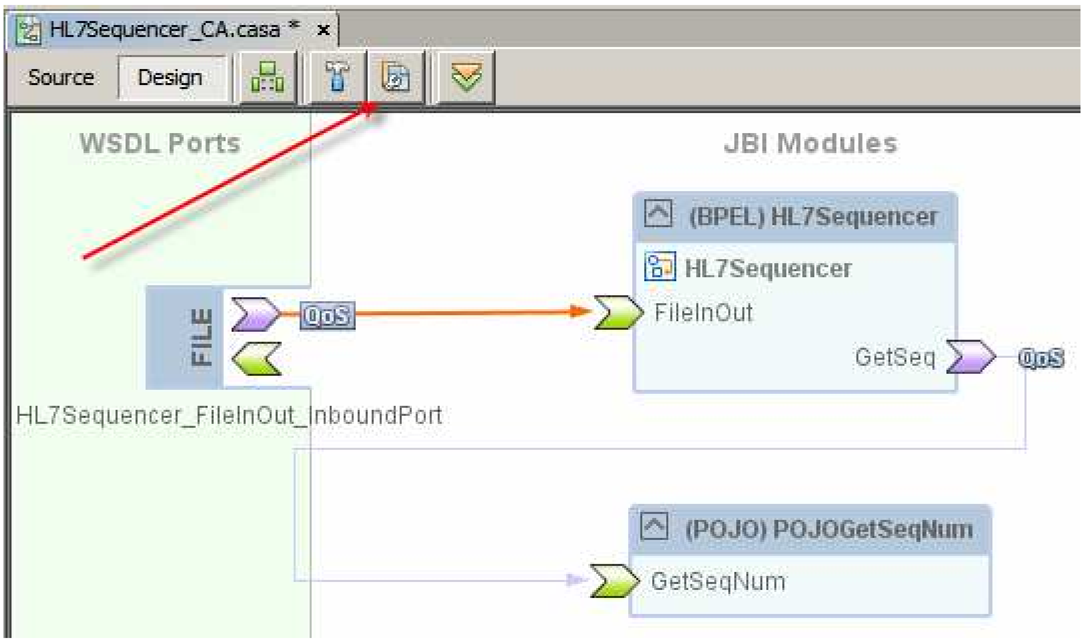
HL7Sequencer_FileInOut_InboundPort -> FileI...

Consumer	
Service Name	{http://j2ee.netbeans.org/...
Endpoint Name	HL7Sequencer_FileInOut_I...
Provider	
Service Name	{http://enterprise.netbeans...
Endpoint Name	FileInboundPortTypeRole_...
Redelivery Extension	
Max Attempts	
Wait Time	
On Failure	
Throttling Extension	
Max Concurrency Limit	1

Max Concurrency Limit

Maximim concurrency limit

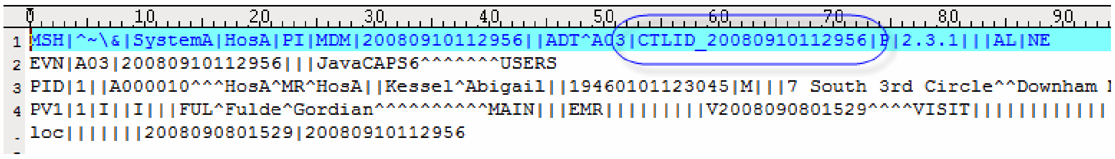
Deploy the project.



Archive containing test messages, ADT_A03_raw.zip, can be downloaded from http://mediacast.sun.com/users/Michael.Czapski-Sun/media/ADT_A03_raw.zip/details.

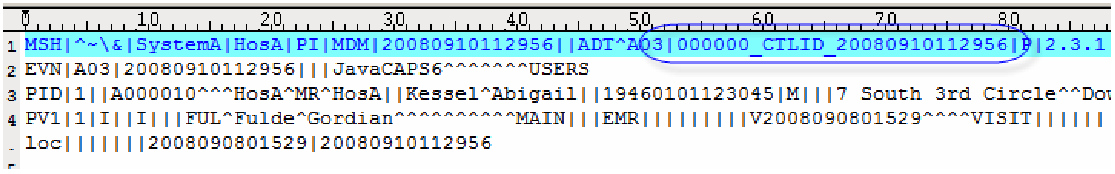
Submit the file, ADT_A03_raw_1.hl7, containing 1 HL7 message.

Inspect the input file content.



```
MSH|^~\&|SystemA|HosA|PI|MDM|20080910112956||ADT^A03|CTLID_20080910112956|P|2.3.1|||AL|NE
EVN|A03|20080910112956|||JavaCAPS6^^^^^^^USERS
PID|1||A000010^^^HosA^MR^HosA||Kessel^Abigail||19460101123045|M|||7 South 3rd Circle^^Downham
PV1|1|I||I|||FUL^Fulde^Gordian^^^^^^^^^^MAIN|||EMR||||||||V2008090801529^^^^VISIT||||||||||
loc||||||2008090801529|20080910112956
```

Inspect the output file content.



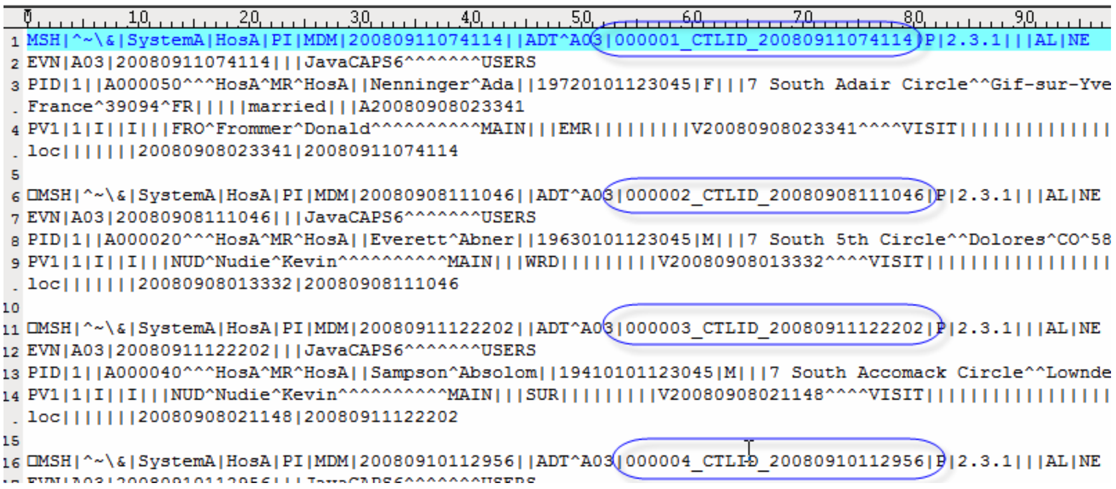
```
MSH|^~\&|SystemA|HosA|PI|MDM|20080910112956||ADT^A03|000000_CTLID_20080910112956|P|2.3.1
EVN|A03|20080910112956|||JavaCAPS6^^^^^^^USERS
PID|1||A000010^^^HosA^MR^HosA||Kessel^Abigail||19460101123045|M|||7 South 3rd Circle^^Dow
PV1|1|I||I|||FUL^Fulde^Gordian^^^^^^^^^^MAIN|||EMR||||||||V2008090801529^^^^VISIT|||||
loc||||||2008090801529|20080910112956
```

Sequence number, 000000, was prepended to the MSH-10 content.

Delete, or move elsewhere, the output file. If you don’t do that records from the next run will be appended to that file, as requested in the File BC configuration.

Submit a file with 30 HL7 messages, ADT_A03_raw_30.hl7, and inspect the content.

Inspect the output file and note that sequence numbers were prepended and started from 000001, since the one record file was processed before.



```
MSH|^~\&|SystemA|HosA|PI|MDM|20080911074114||ADT^A03|000001_CTLID_20080911074114|P|2.3.1|||AL|NE
EVN|A03|20080911074114|||JavaCAPS6^^^^^^^USERS
PID|1||A000050^^^HosA^MR^HosA||Nenninger^Ada||19720101123045|F|||7 South Adair Circle^^Gif-sur-Yve
France^39094^FR|||||married|||A20080908023341
PV1|1|I||I|||FRO^Frommer^Donald^^^^^^^^^^MAIN|||EMR||||||||V20080908023341^^^^VISIT||||||||||||||
loc||||||20080908023341|20080911074114

□MSH|^~\&|SystemA|HosA|PI|MDM|20080908111046||ADT^A03|000002_CTLID_20080908111046|P|2.3.1|||AL|NE
EVN|A03|20080908111046|||JavaCAPS6^^^^^^^USERS
PID|1||A000020^^^HosA^MR^HosA||Everett^Abner||19630101123045|M|||7 South 5th Circle^^Dolores^CO^58
PV1|1|I||I|||NUD^Nudie^Kevin^^^^^^^^^^MAIN|||WRD||||||||V20080908013332^^^^VISIT||||||||||||||||
loc||||||20080908013332|20080908111046

□MSH|^~\&|SystemA|HosA|PI|MDM|20080911122202||ADT^A03|000003_CTLID_20080911122202|P|2.3.1|||AL|NE
EVN|A03|20080911122202|||JavaCAPS6^^^^^^^USERS
PID|1||A000040^^^HosA^MR^HosA||Sampson^Absolom||19410101123045|M|||7 South Accomack Circle^^Lownde
PV1|1|I||I|||NUD^Nudie^Kevin^^^^^^^^^^MAIN|||SUR||||||||V20080908021148^^^^VISIT||||||||||||||||
loc||||||20080908021148|20080911122202

□MSH|^~\&|SystemA|HosA|PI|MDM|20080910112956||ADT^A03|000004_CTLID_20080910112956|P|2.3.1|||AL|NE
```

JVM global, ephemeral sequencing works.

To reset the sequence to zero re-deploy the project. This is why this kind of sequencing is called ephemeral (short lived - http://www.google.com.au/search?q=define%3Aephemeral).

The private static int, which holds the current sequence, will be destroyed if the application server is restarted, application is undeployed or redeployed.

Summary

This note walked through development of a JVM global, ephemeral sequence number generator, in a form of a POJO useable from a BPEL process. To demonstrate the use of the sequence generator a BPEL project, enriching HL7 v2.3.1 messages with sequence numbers prepended to the MSH-10 filed, was also developed and exercised. This kind of project can be used, as I have done, to prepare data for HA testing, and other occasions where message processing in sequence is required and needs to be confirmed.

References

1 "GlassFish ESB v2.2 Field Notes - Exercising Load Balanced, Highly Available, Horizontally Scalable HL7 v2 Processing Solutions", Accessed: January 9, 2010, Available: http://blogs.sun.com/javacapsfieldtech/entry/glassfish_esb_v2_2_field1