Oracle SOA Suite 11g R1 PS5

SOA Suite for healthcare integration Series

HL7 v2 Inbound CMM to Outbound Pass-through Solution

michael@czapski.id.au

December 2012

Table of Contents

Introduction 1

Solution Overview 2

HL7 v2 Inbound (Receiver) 4

HL7 v2 Outbound (Sender) 5

Configure ADT Sender 5

Prepare to receive ADT messages 9

HL7 v2 Outbound pass-through 9

Develop Router Composite 9

Send ADT messages 19

Explore Message Tracking 20

HL7 v2 Inbound to Outbound with XML translation 21

Configure Translation and Functional Acknowledgement in Inbound Endpoint 21

Configure Translation and Functional Acknowledgement in Outbound Endpoint 22

Re-configure and re-deploy the Router project 22

Process ADT messages 24

Explore Message Tracking 25

Summary 28

Introduction

The archetypical "Hello World" solution in the HL7 v2 messaging world will consist of a HL7 v2 message receiver which writes the messages it receives to files in the file system. A more sophisticated solution will receive an inbound message and send it out to an external system, possibly transforming it before sending.

This article works through the mechanics of configuring the "SOA Suite for healthcare integration" to send a HL7 v2.3.1 ADT message as a Canonical Message, extending the solution developed in the previous article, "" to be found at .

This article assumes that the reader is sufficiently familiar with HL7 v2 and HL7 v2 messaging to require no elaboration on the message structures, message

acknowledgement processing and the "equivalence" of HL7 v2 delimited and HL7 v2 XML message forms.

This article assumes that the reader has the SOA Suite for healthcare integration environment with all necessary components installed and ready to use. The Bill of Materials for such an environment and a discussion on where the components can be obtained is provided in the earlier article, "SOA Suite for healthcare integration Series - Overview of the Development Environment", to be found at http://blogs.czapski.id.au/2012/08/soa-suite-for-healthcare-integration-series-overview-of-the-development-environment.

This article assumes that the reader completed the solution discussed in the earlier article, "SOA Suite for healthcare integration Series – HL7 v2 Inbound to File Solution", to be found at http://blogs.czapski.id.au/2012/11/soa-suite-for-healthcare-integration-series-hl7-v2-inbound-to-file-solution.

Solution Overview

An enterprise system, say a Hospital Information System, a Patient Administration System, or some other system in a Hospital, produces HL7 v2 ADT messages, specifically ADT A01 – Admission, ADT A03 – Discharge and ADT A08 – Update Patient Information messages. These messages will be cast to the Canonical Message Model using the CMM message structure which was developed in the earlier article, "SOA Suite for healthcare integration Series - Creating a Canonical HL7 v2 Message Model", to be found at http://blogs.czapski.id.au/2012/09/soa-suite-for-healthcare-integration-series-creating-a-canonical-hl7-v2-message-model, then sent out through an outbound HL7 endpoint to an external system as the original HL7 v2 delimited messages.

The inbound SOA Suite for healthcare integration adapter will perform the casting activity while translating the message from HL7 v2 delimited to the "equivalent" XML format. The outbound SOA Suite for healthcare integration adapter will translate the message from HL7 v2 XML to the "equivalent" HL7 v2 delimited format before sending it out.

The runtime components and their relationships are presented in Figure 1.

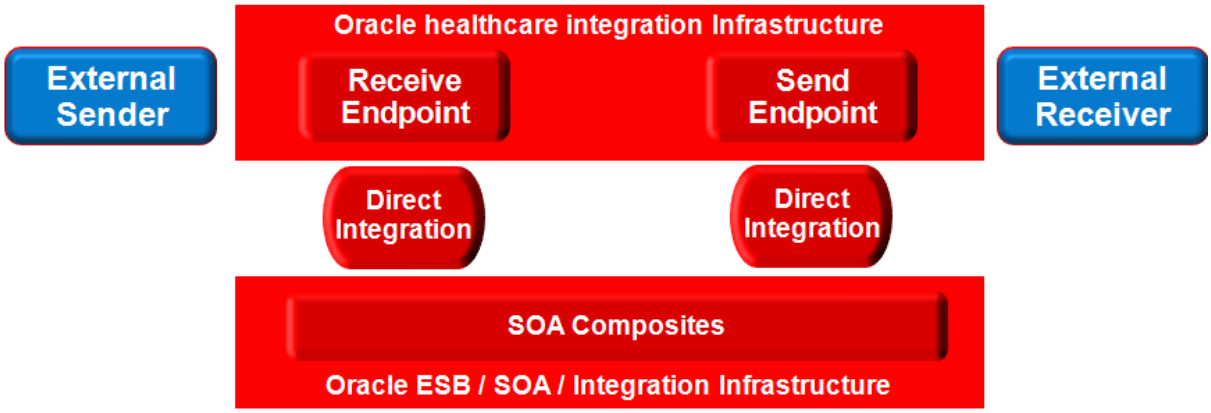


Figure 1 Runtime Components of the Solution

To summarise:

External Sender is the component which stands for a HIS or PAS – the sender of HL7 messages

The Oracle Healthcare Integration infrastructure is the part of the SOA Suite which deals with HL7 messages, acknowledgements, message tracking, message persistence, message processing KPI collection and so on, and the Receive Endpoint is the listening endpoint which receives messages

Direct Integration is the behind-the-scenes mechanism which hands messages over to an appropriate SOA Suite-based logic component for further processing or hands over messages produced by SOA Suite components to the healthcare integration infrastructure for processing (translation, sending, tracking).

ESB / SOA / Integration infrastructure hosts the SOA Composites and other logic components which process messages, whether HL7 v2 or not.

The solution components are depicted in Figure 2.

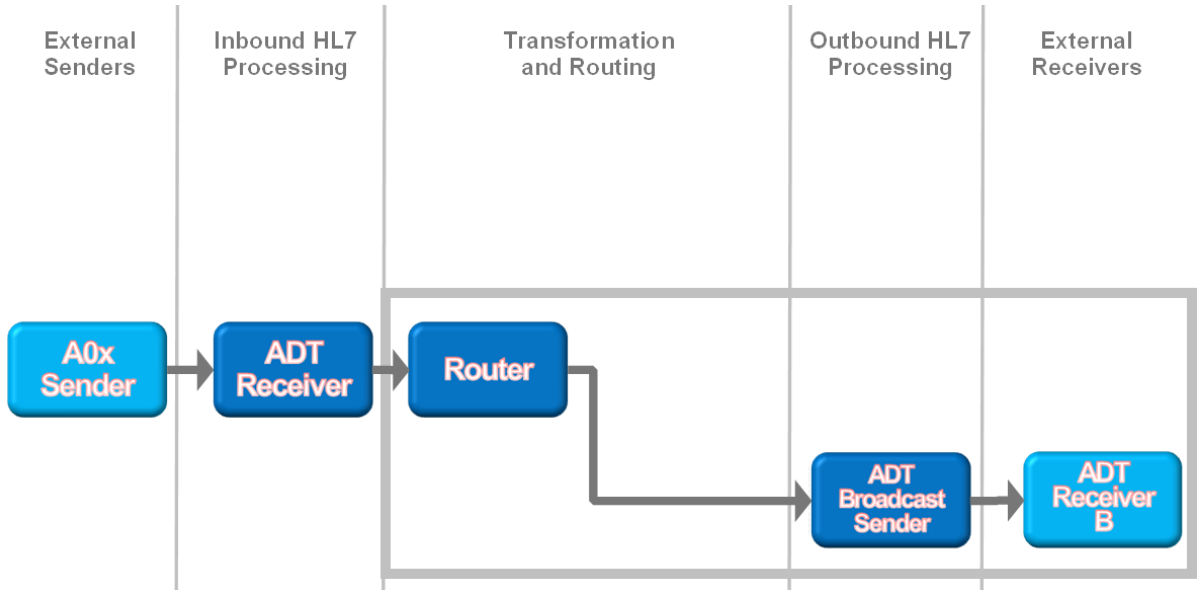


Figure 2 Solution Components

The diagram uses the convention which clearly separates the external systems, the SOA Suite for healthcare integration-specific components and generic SOA Suite components using the "swim-line" analogy.

A0x Sender is the CMDHL7 sender tool, or another tool capable of sending HL7 v2 Delimited messages over TCP/IP using the MLLP protocol. It will send A01 and A03 messages.

ADT Receiver is the SOA Suite for healthcare integration HL7 v2 listener.

Router is a SOA Composite which receives the message from the HL7 listener and "routes" it to the HL7 sender.

ADT Broadcast Sender is the SOA Suite for healthcare integration HL7 v2 sender.

ADT Receiver B is the CMDHL7 listener tool, or another tool capable of receiving HL7 v2 Delimited messages over TCP/IP using the MLLP protocol and acknowledging them. It will receive A01 and A03 messages.

The solution will receive a HL7 v2 message, acknowledge it with an immediate acknowledgement and pass it onto the SOA Composite. The SOA Composite will pass it to the HL7 sender for sending out, without transforming it in any way. The immediate

acknowledgement will be sent as soon as the message is received and persisted, before it is processed in any way. The acknowledgement received from the HL7 listener will be discarded.

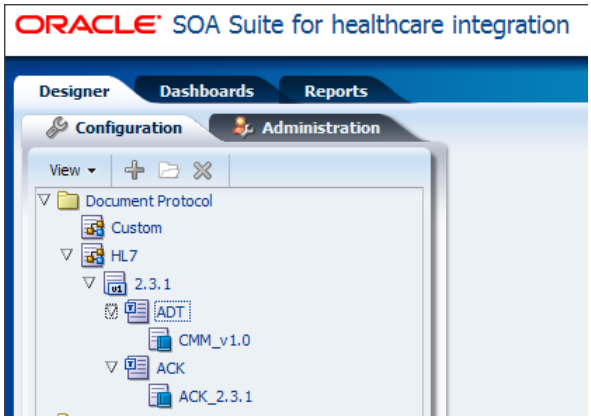
HL7 v2 Inbound (Receiver)

It is assumed that the WebLogic Server is running, as it needs to be, to allow us to interact with the SOA Suite for healthcare integration infrastructure.

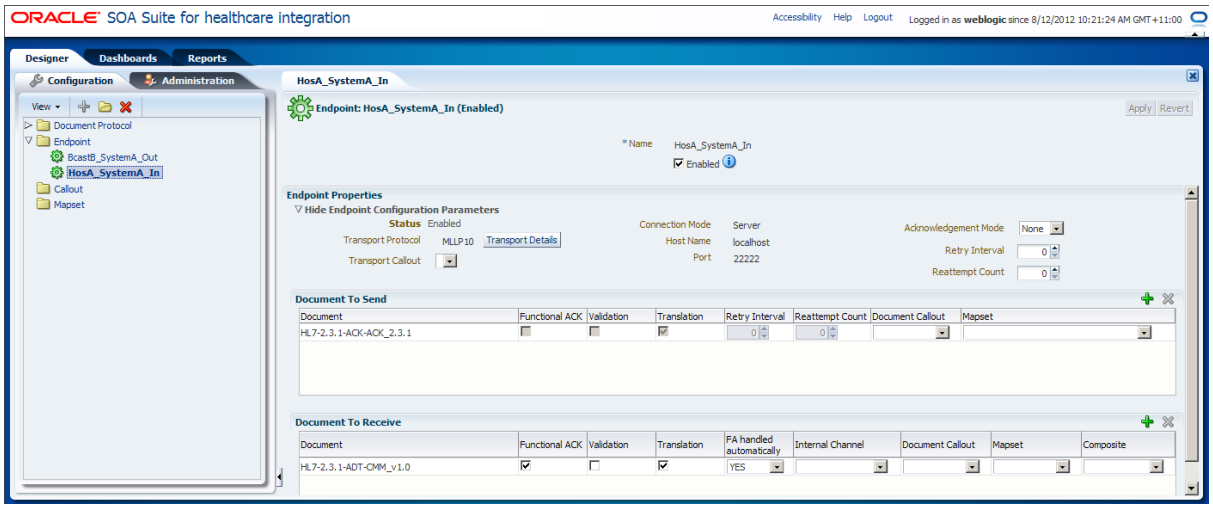
Here we will verify that the receiver endpoint, "HosA_SystemA_In", exists and will reconfigure it so that it does not perform translation from HL7 v2 delimited to HL7 v2 XML format on the way in. This is so that we can demonstrate HL7 v2 Delimited In to HL7 v2 Delimited Out pass-through functionality. We will change that later to perform translation for mapping.

☐ Start Healthcare Integration Console, http://localhost:7001/healthcare, and log in

☐ Expand "Designer"→"Configuration"→"Document Protocols"→"HL7"→"2.3.1" "ADT" and "ACK" hierarchies, making sure that the "CMM_v1.0" and the "ACK_2.3.1" document exist

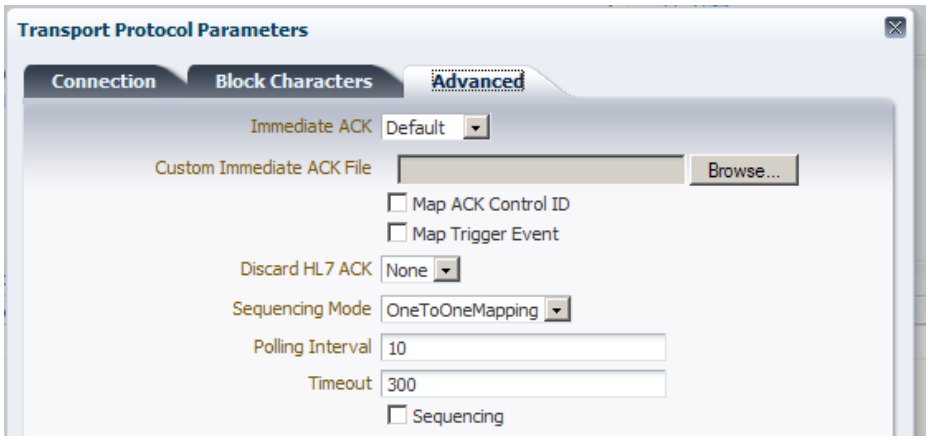


☐ Expand "Designer"→"Configuration"→"Endpoints" hierarchy, making sure that the endpoint "HosA_SystemA_In" exists and is enabled

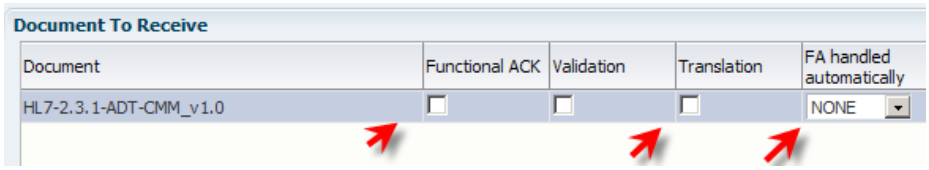


☐ Click the "Transport Details" button, click the "Advanced" Tab, choose "Default" for the

"Immediate ACK" setting then click "OK" to complete the dialogue



☐ Uncheck the "Translation" checkbox, uncheck the "Functional ACK" checkbox and change the "FA handled automatically" to NONE in "Document To Receive" part of the configuration panel and click "Apply" to apply changes



This completes the re-configuration of the receiver for the time being.

HL7 v2 Outbound (Sender)

In this section we will configure the sender endpoint, which will send HL7 v2 Delimited messages to the external receiver.

Configure ADT Sender

The sender endpoint will send messages of the CMM_v1.0 type to the external system.

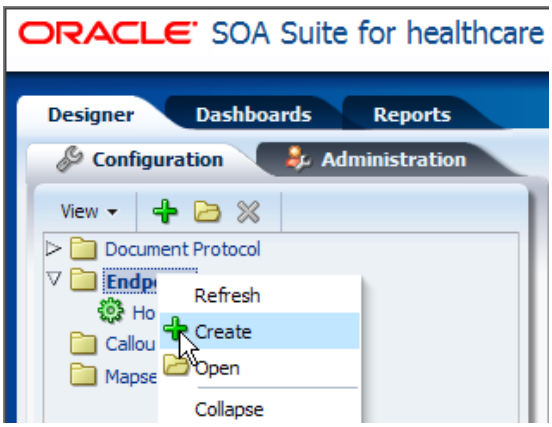
The sender will be a TCP/IP sender supporting the MLLP 1.0 encapsulation protocol. Such a sender establishes a connection to a listener which listens on a specific TCP/IP port, sends messages to it, receives acknowledgements back over the same connection, and processes them until either party disconnects.

In the sending mode the endpoint is configured as a "client".

☐ Start the Healthcare Integration Console application in your favorite web browser – http://localhost:7001/healthcare.

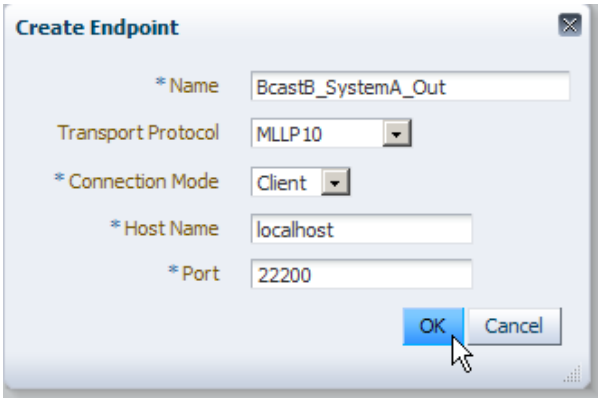
☐ Log in with administrative credentials, for example weblogic/welcome1.

☐ Right-click the "Endpoint" node in the "Configuration" tab and choose "Create"



☐ Enter the following in the "Configure Endpoint" dialogue box then click "OK"

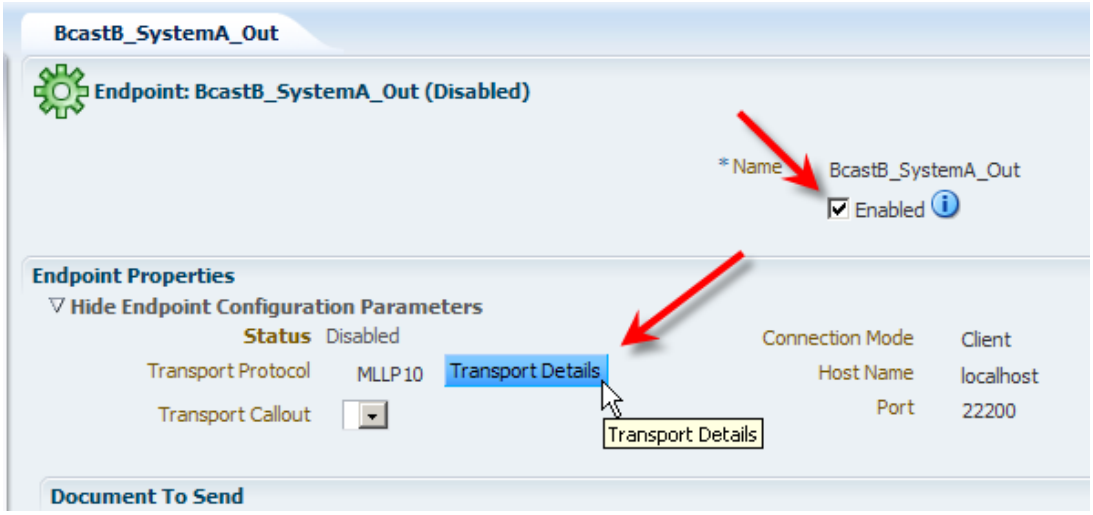
- Name: BcastB_SystemA_Out
- Transport Protocol: MLLP10
- Connection Mode: client
- Host Name: localhost (or the name of whatever host you are using)
- Port: 22200



The endpoint is not quite configured as we want it. We will change the non-default values to suit our requirement in the following steps.

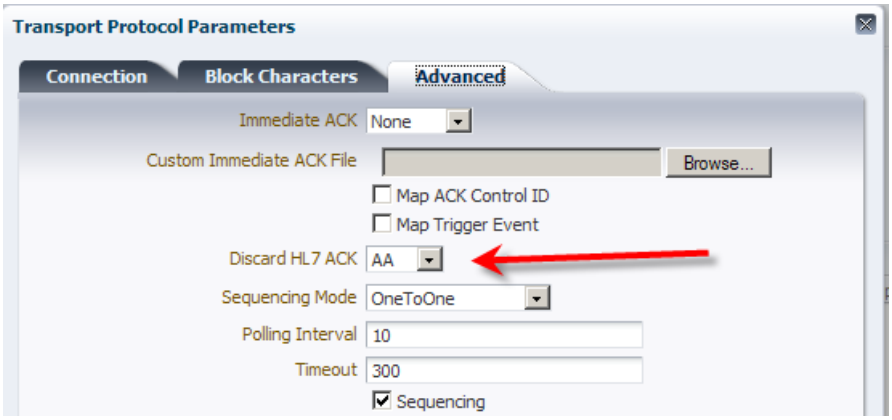
☐ Check the "Enabled" checkbox. When we "Apply" this configuration later the endpoint will be started.

☐ Click the "Transport Details" button

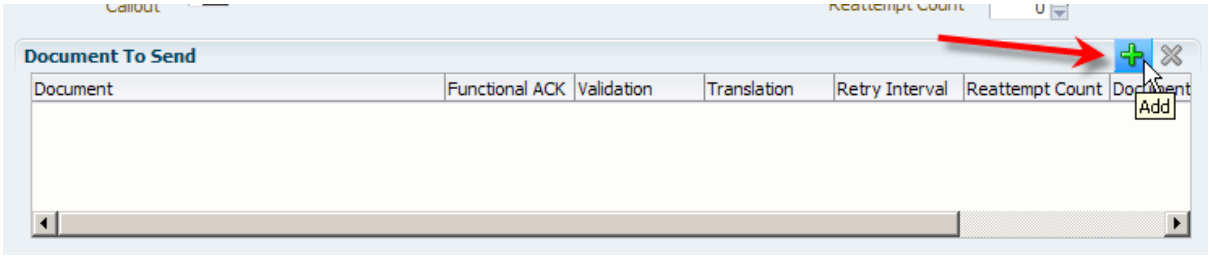


☐ Click the "Advanced" tab in the "Transport Protocol Parameters" dialogue box, set the following properties, and click "OK":

- Discard ACK: AA

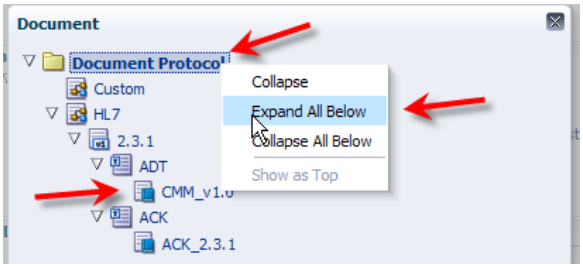


☐ Click the "Add" "button" (a plus sign) in the "Documents to Send" section

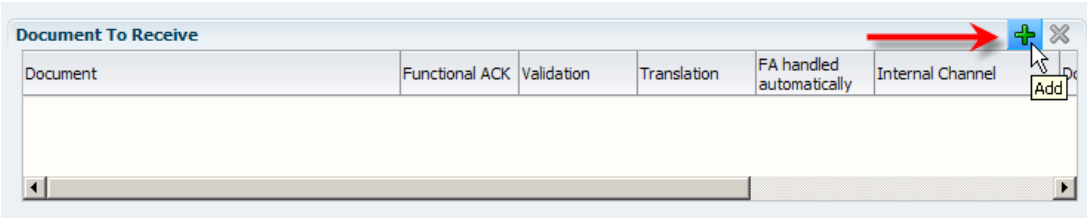


☐ Right-click the "Document Protocol" node in the "Document" dialog box and choose "Expand All Below"

☐ Select the "CMM_v1.0" document in the HL7→2.3.1→ADT hierarch and click "OK"

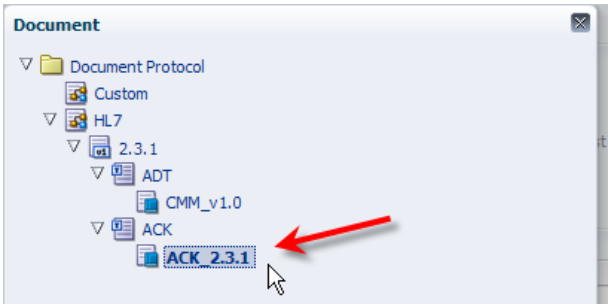


☐ Click the "Add" "button" (a plus sign) in the "Documents to Receive" section

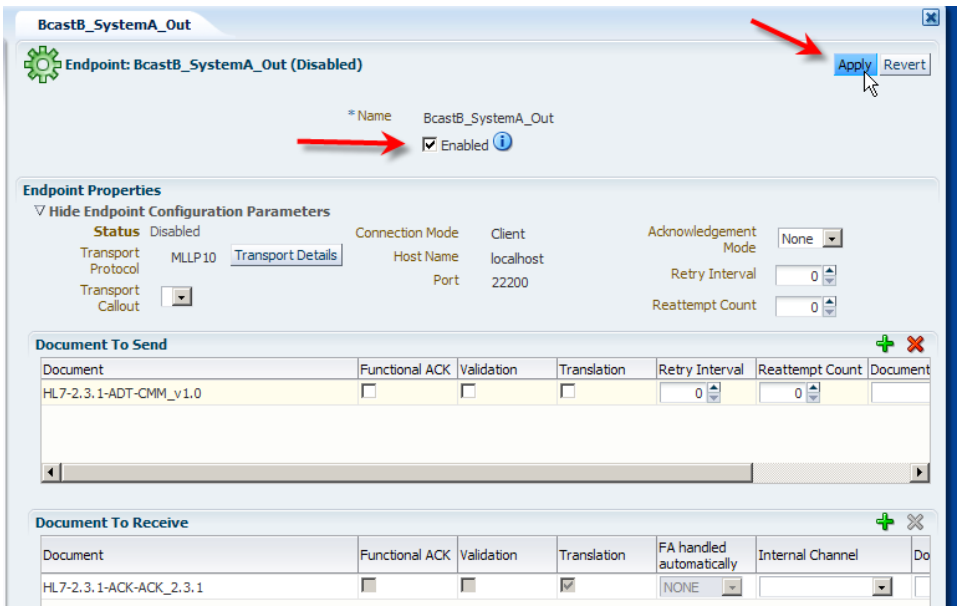


☐ Right-click the "Document Protocol" node in the "Document" dialog box and choose "Expand All Below"

☐ Select the "ADT_2.3.1" document in the HL7→2.3.1→ACK hierarch and click "OK"



☐ Uncheck the "Translation" checkbox, review the configuration to make sure it is correct and click the "Apply" button, remembering that with the "Enabled" checkbox checked this action will cause the SOA Suite for healthcare integration to attempt to start the endpoint



Prepare to receive ADT messages

We will use the CMDHL7Listener command line client to receive HL7 ADT messages look at the output in the output directory specified on the listener's command line – for me c:\hl7\received. The CMDHL7Listener will display trace of message exchange in the console window. The SOA Suite for healthcare integration will record message tracking information which we will look at a later stage.

Please note that in this solution the CMDHL7Listener returns an ACK as soon as it gets the message.

☐ Check that your configured output directory is empty

☐ In a command / terminal window execute the following command

```
java -jar c:\tools\CMDHL7\CMDHL7Listener_v0.7.jar -c ID_ -p 22200 -s
c:\hl7\received
```

☐ Inspect the CMDHL7Listener console output making sure the listener started and is listening on the appropriate port

```
08/12/2012 10:45:36 AM au.id.czapski.hl7.CMDHL7Listener main
INFO: Port: 22200
08/12/2012 10:45:36 AM au.id.czapski.hl7.CMDHL7Listener main
INFO: Store in: c:\hl7\received
08/12/2012 10:45:36 AM ca.uhn.log.HapiLogImpl info
INFO: au.id.czapski.hl7.SimpleACKApplication registered to handle *^*
messages
08/12/2012 10:45:36 AM ca.uhn.log.HapiLogImpl info
INFO: SimpleServer running on port 22200
```

HL7 v2 Outbound pass-through

Unlike in the previous article, in this article, the term "Router" is used as a descriptive term. IN this segment of the article, the SOA Composite called the "Router" will "route" each message it receives to an external HL7 receiver via a HL7 Sender endpoint, without changing it.

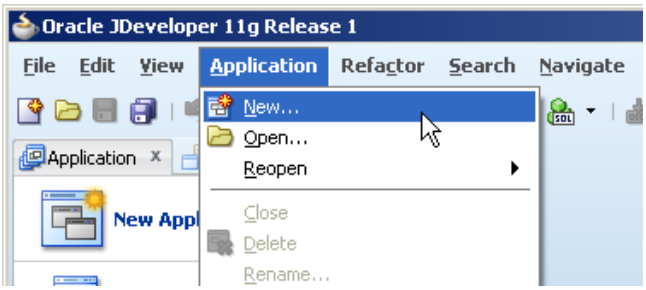
We made sure that messages are not translated by the ADT Receiver endpoint by unchecking the "Translation" checkbox in the endpoint configuration. We also made sure that the sending endpoint does not expect XML messages by unchecking the "Translate" checkbox in its configuration. As SOA Suite adapters by default deal with XML messages, we need to make sure we override default configuration of the ADT Receiver and ADT Sender endpoints.

Develop Router Composite

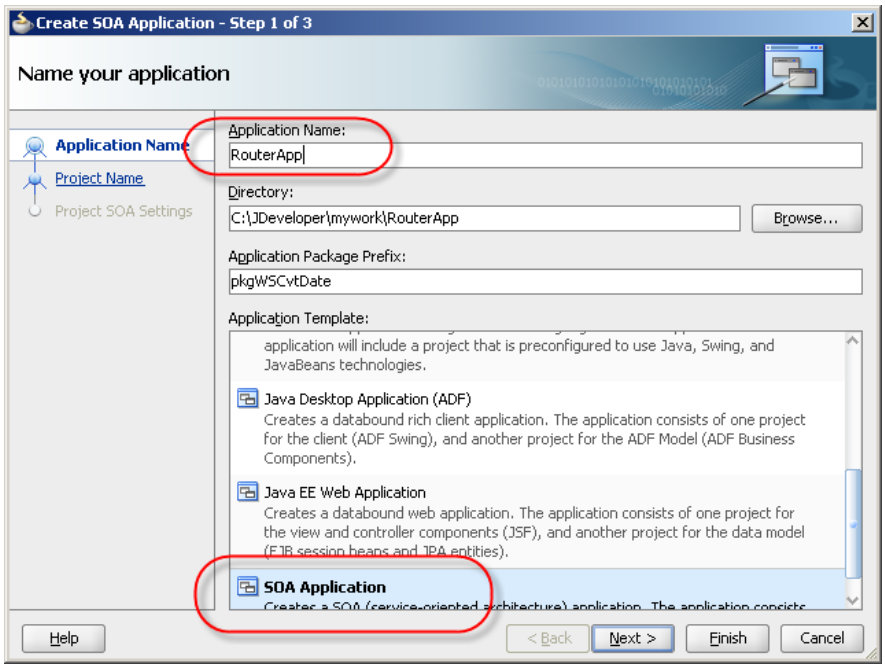
Tying together two adapters in SOA Suite requires a logic component - BPEL, Mediator, Business Rules, … . The role of this component is to receive a message from the inbound adapter and pass it to the outbound adapter, potentially transforming it in the process. At this point in the solution development we will simply pass the message as is, therefore the logic component will be the simplest we can get – Mediator component – and it will be configured as a pass-through.

☐ Start the JDeveloper Studio IDE

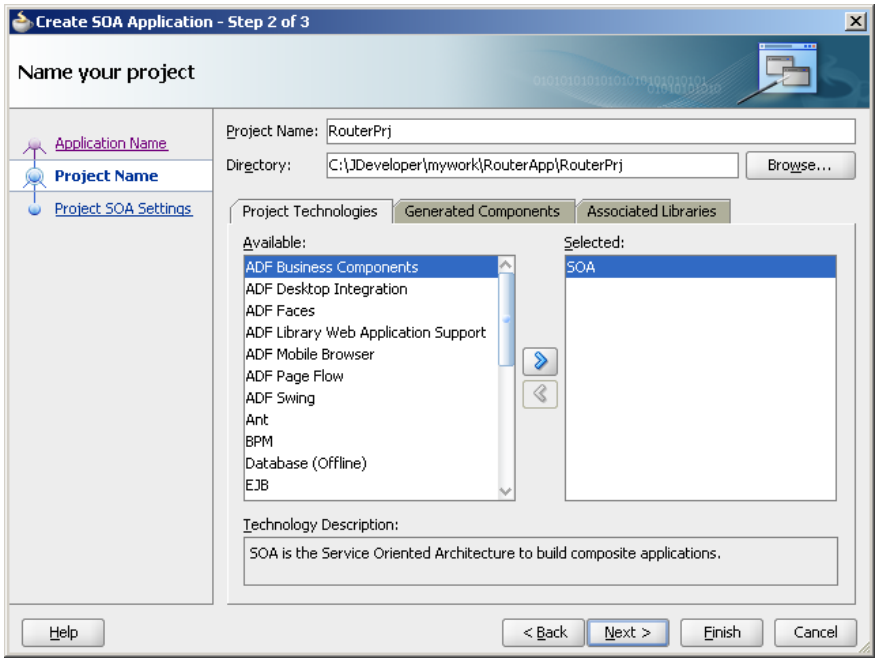
☐ Pull down the "Application" menu and choose "New…"



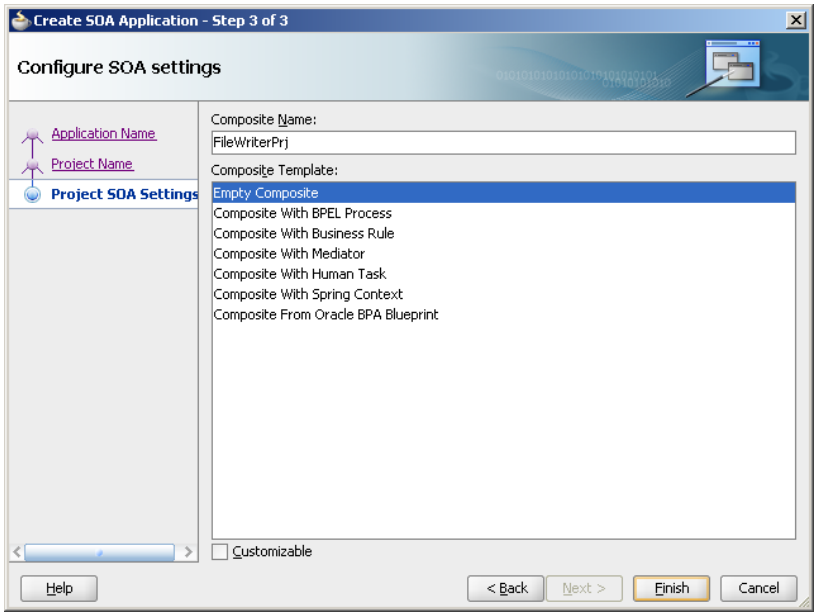
☐ Enter "RouterApp" as "Application Name", choose "SOA Application" and click "Next"



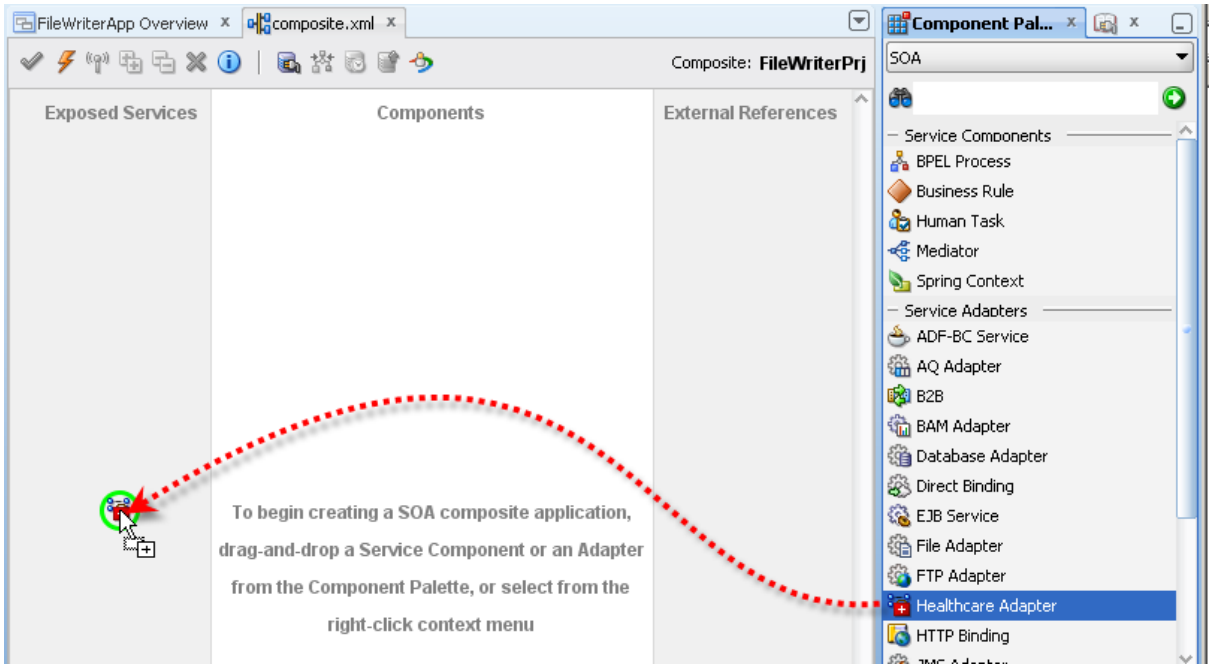
☐ Enter "RouterPrj" as "Project Name" and click "Next"



☐ Accept "Empty Composite" and click "Finish"

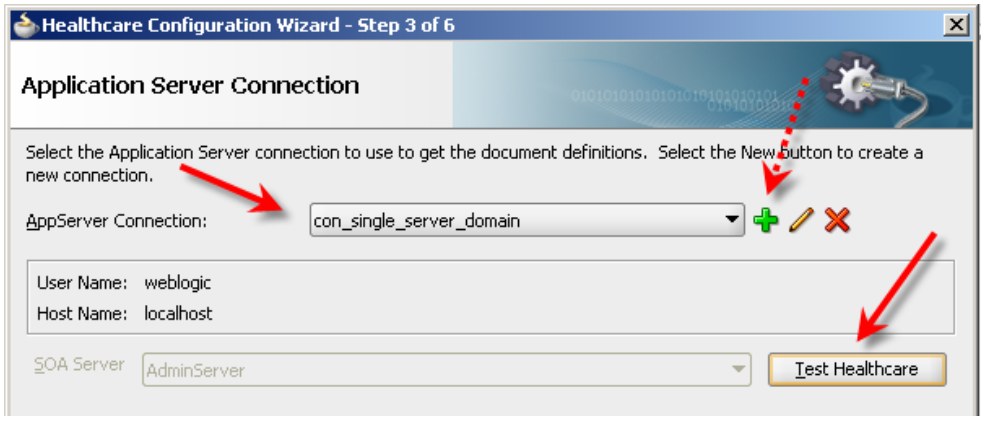


☐ Drag the "Healthcare Adapter" from the list of "Service Adapters" to the "Exposed Services swim-line and release



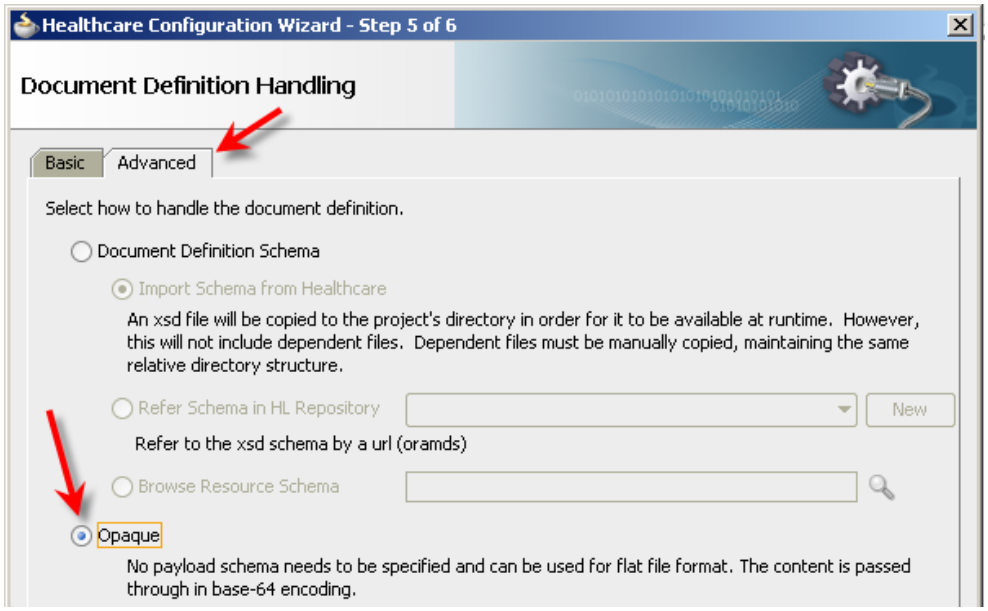
☐ Name the adapter "HCI_ In" and click "Next"

☐ Choose your "AppServer Connection" from the drop down or add one with the "plus" button if you have not done so before, then click the "Test Healthcare" button to make sure JDeveloper and the appropriate WebLogic server can communicate, then click "Next"

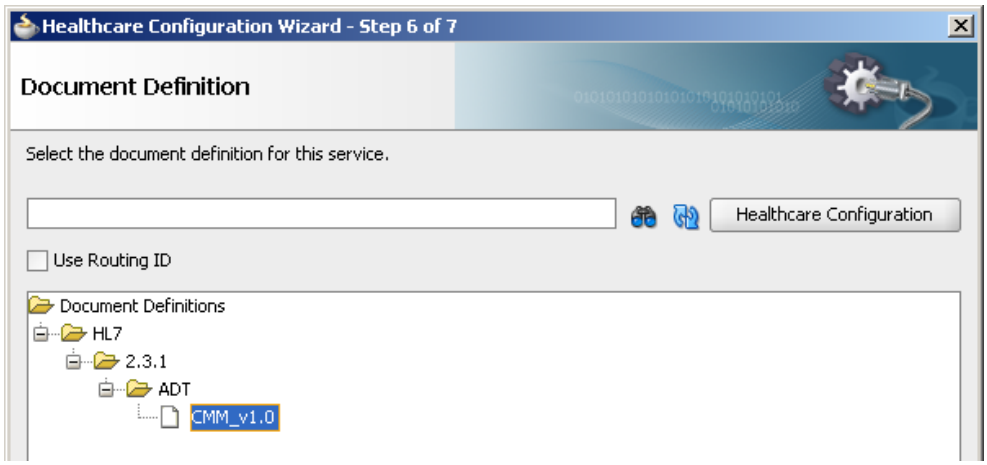


☐ Select the "Receive" operation and click "Next"

☐ In the "Document Definition Handling" dialogue box click the "Advanced" tab, select "Opaque" and click "Next" – this is where we are instructing the SOA Suite leave the message alone and not try to treat is an a XML message



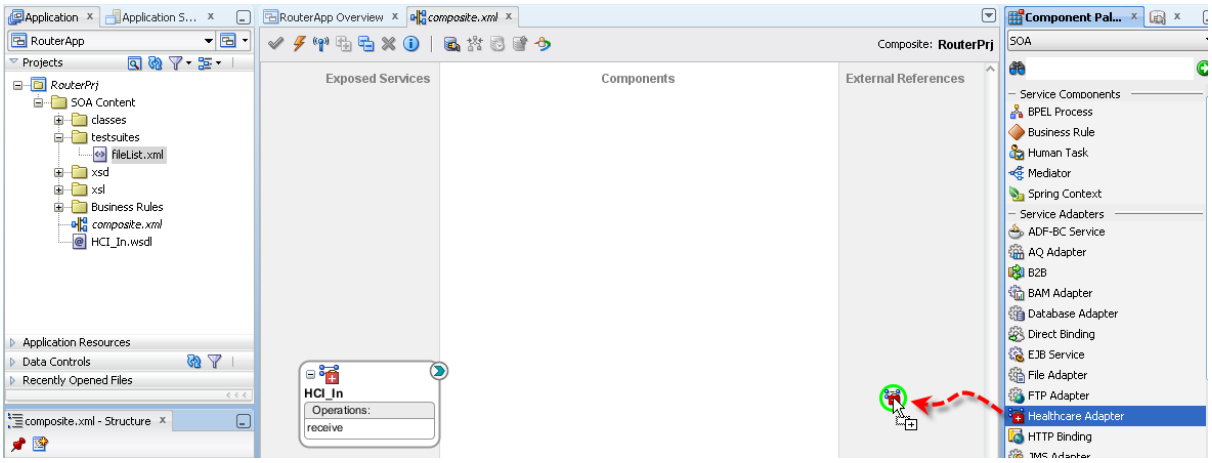
☐ Expand the Document Protocol hierarchy, select the "CMM_V1.0" document, click "Next" and "Finish"



The Healthcare Adapter is configured to receive untranslated HL7 messages from the ADT Receiver endpoint. The two work in concert – one receives and acknowledges messages and the other makes them available for processing by a SOA Composite.

Let's now add and configure the Outbound Healthcare Adapter.

☐ Drag the "Healthcare Adapter" from the "Service Adapters" to the "External References" swim-line



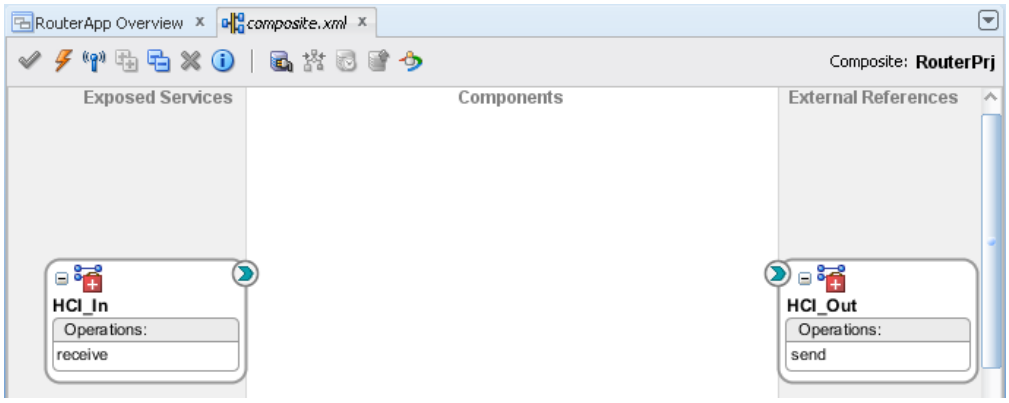
☐ Name it "HCI_Out" and click "Next"

☐ Choose AppServer Connection and click "Next"

☐ Choose "Send" operation and click "Next"

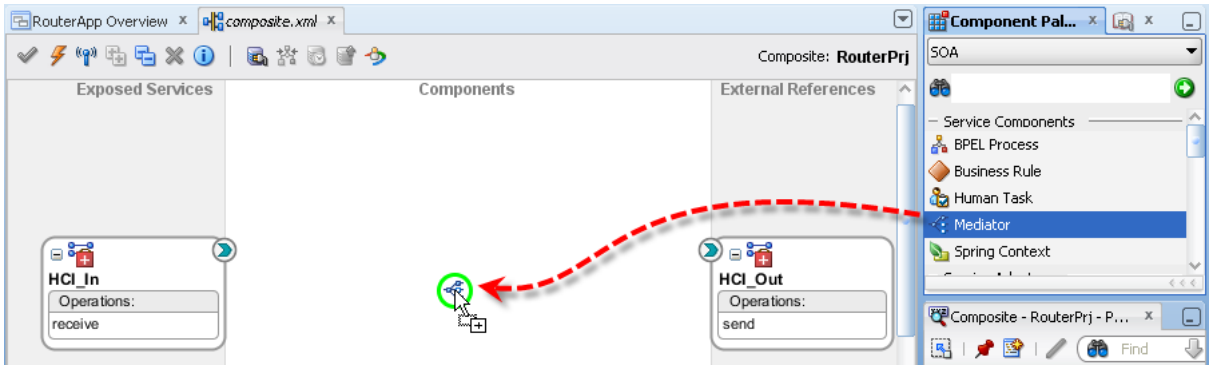
☐ Click the "Advanced" Tab, choose "opaque" and click "Next"

☐ Expand the "Document Definitions" tree, choose "CMM_v1.0" then click "Next" and "Finish"



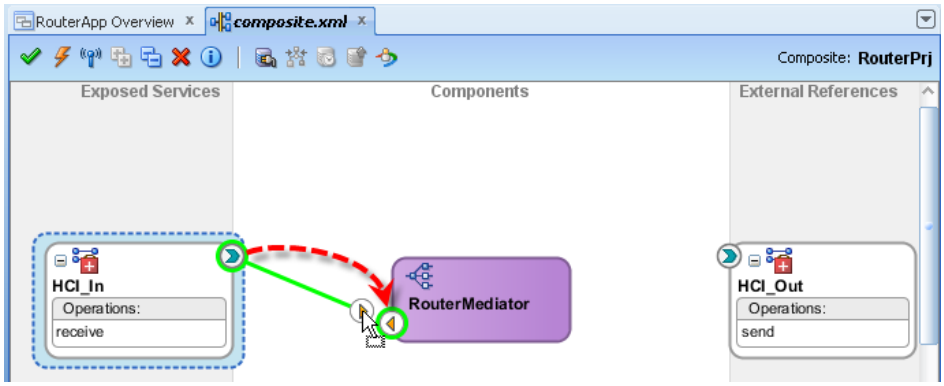
The outbound healthcare adapter is configured. We now need to tie the inbound and the outbound with a "Service Component". We will use "Mediator" component for simplicity.

☐ Drag the "Mediator" component from the list of "Service Components" to the "Components" swim-line

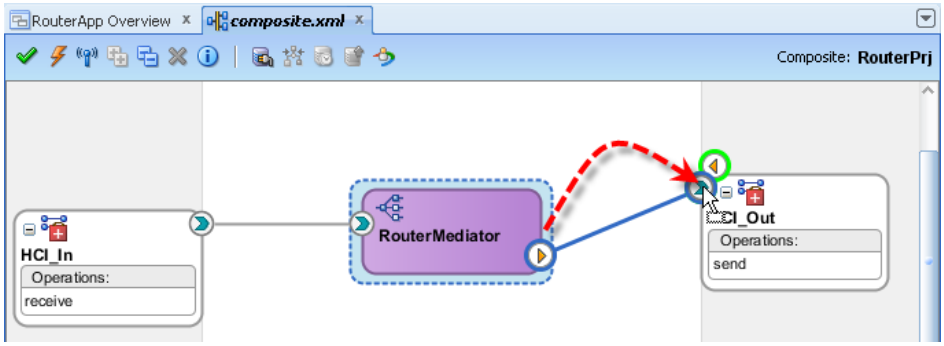


☐ Name the component "RouterMediator" and click "OK"

☐ Drag from the "Chevron in a circle" symbol in the top right hand corner of the HCI_In adapter to the "RouterMediator" component's left pointing triangle to connect the two components



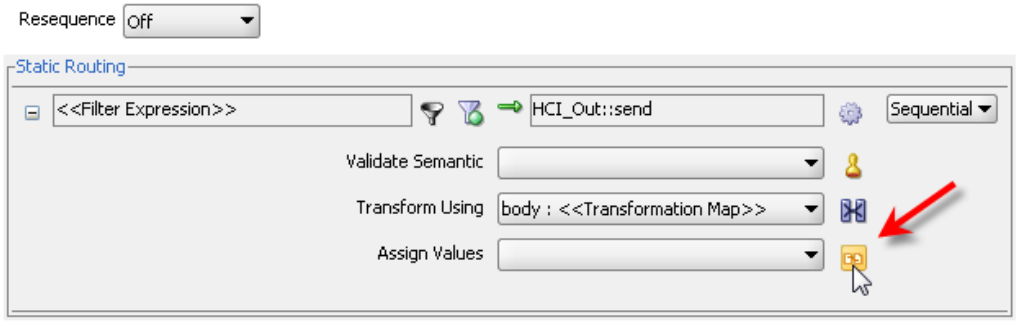
☐ Drag from the right hand pointing triangle in the "RouterMediator" to the chevron in a circle symbol in the top left corner of the "HCI_Out" adapter to connect the two



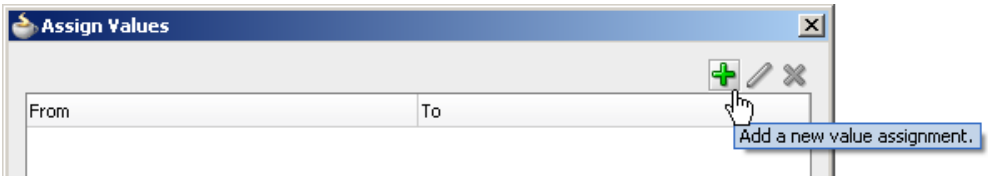
If we were passing through XML messages we would be done configuring the Mediator component. By default it will simply pass its input to its output. Because we are using opaque messages we must explicitly configure this pass through. We must also configure specific properties which will allow the SOA Suite for HealthCare integration outbound infrastructure to locate the correct outbound adapter, hence to send messages to the correct external system.

☐ Double-click the "RouterMediator" component to open its properties

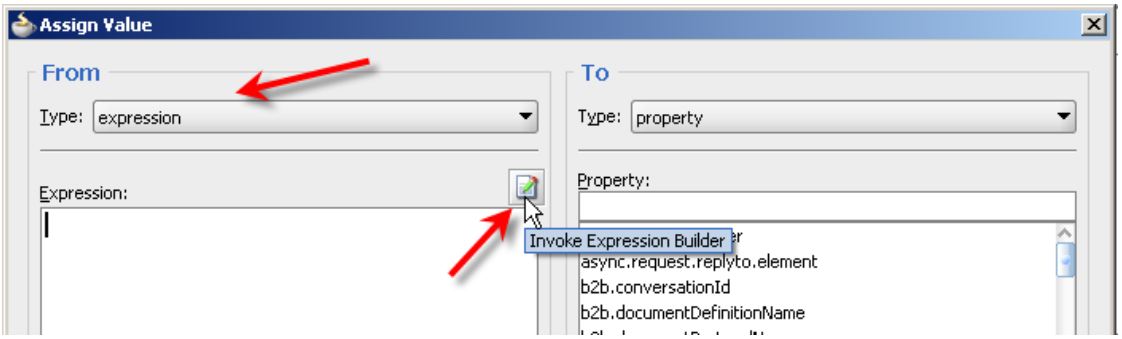
☐ Click on the "Assign Values" button to start assigning values



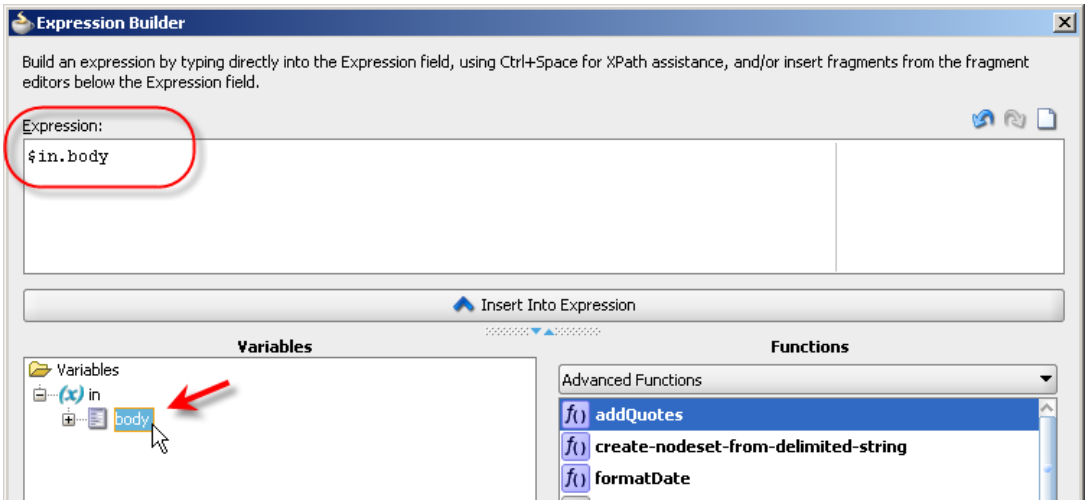
☐ Click the "Add a new value assignment" (the plus sign) button



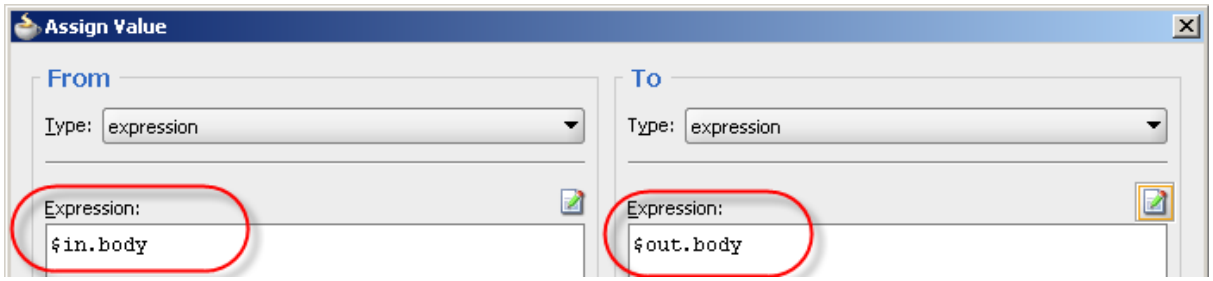
☐ Choose "expression" in the "Type" drop-down on the "From" side and click the "Invoke Expression Builder" button



☐ Expand the "in" variable and double-click the "body" node to make it appear in the "Expression" box – alternatively select the "body" node and click the "Insert Into Expression" button – then click "OK"



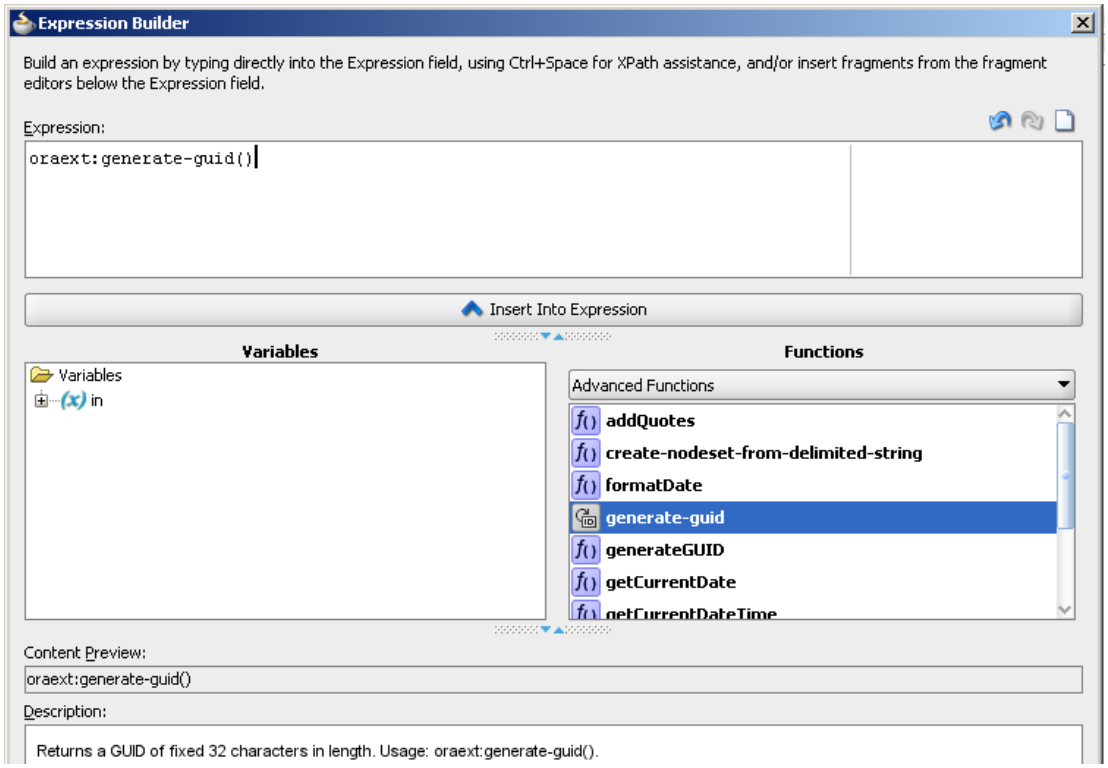
☐ Repeat the process for the "To" side, choosing "expression" for "Type", clicking the "Invoke Expression Builder" and adding the $out.body expression



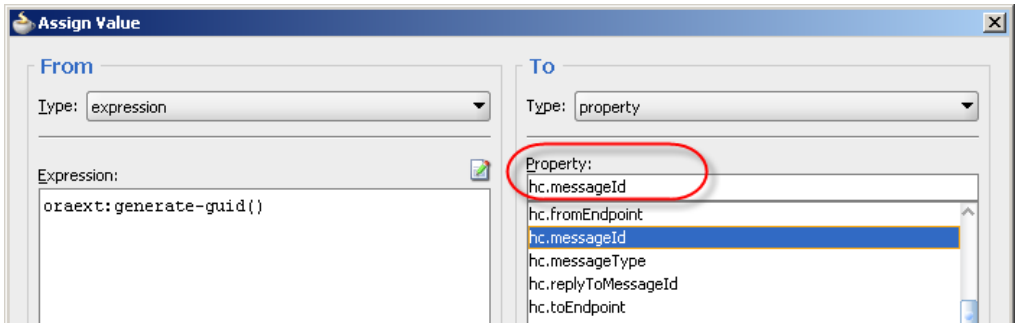
☐ Click "OK" to complete assignment

☐ Click "Add new value assignment." in the "Assign Values" dialogue

☐ Choose "Expression" on the "From" side, invoke "Expression Builder", choose "generate-guid" from the "Advanced Functions" list, add it to the "Expression:" box and click "OK" to close the "Expression Builder"



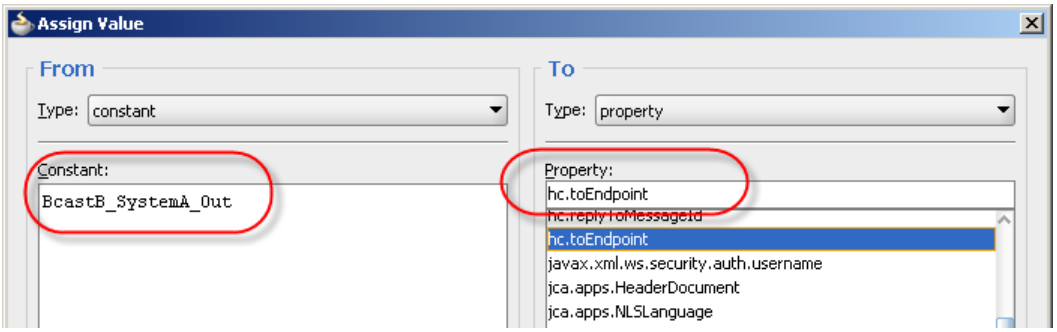
☐ Choose "hc.messageId" property from the properties lisk and click "OK" to close the "Assign Value" dialogue box



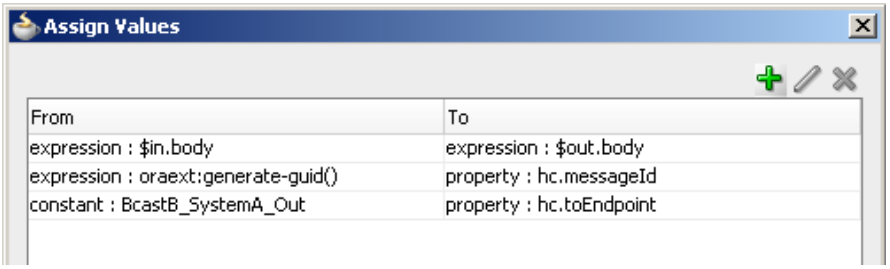
☐ Click "Add new value assignment." in the "Assign Values" dialogue

☐ Choose "Constant" on the "From" side and enter "" into the data entry box

☐ Choose the "hc.toEndpoint" in the properties list on the "To" side and click "OK" to dismiss the dialogue box – the constant value is the name of the sending endpoint as configured in the Healthcare Integration Console

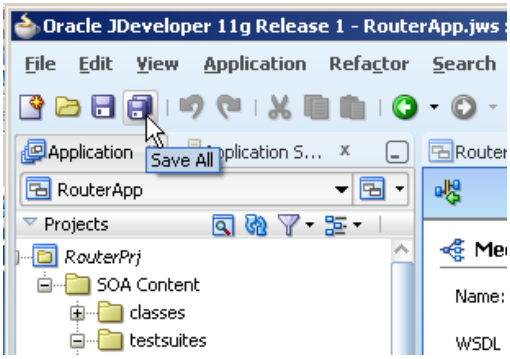


☐ Inspect assignments then click "OK" to complete assignment

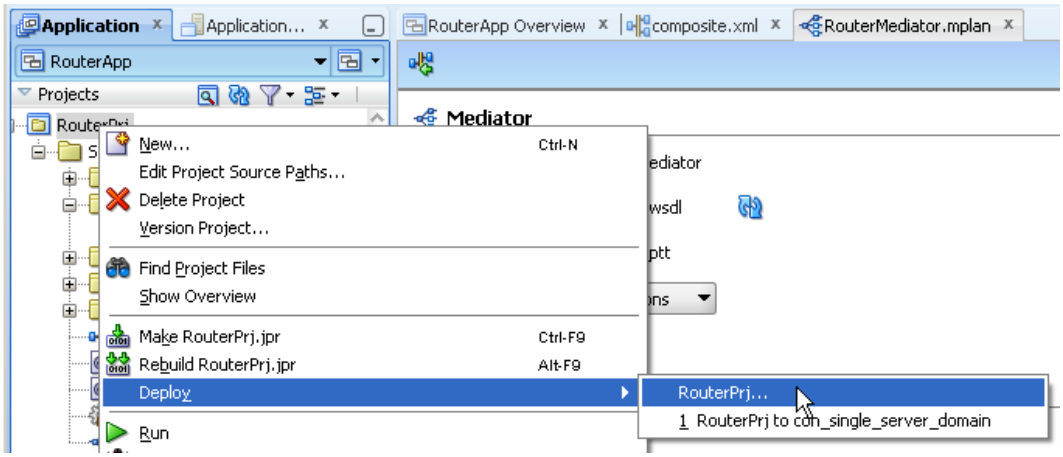


The application development is completed. Let's now save and deploy this application to the nominated application server.

☐ Click the "Save All" button in the main toolbar

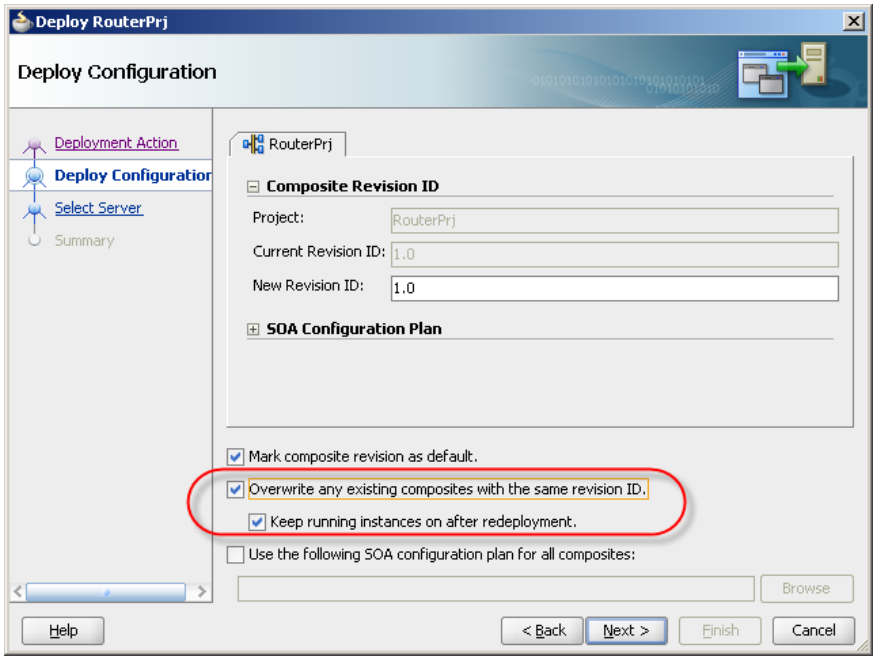


☐ Right-click on the name of the name of the project – "RouterPrj" – choose "Deploy" then "RouterPrj…"



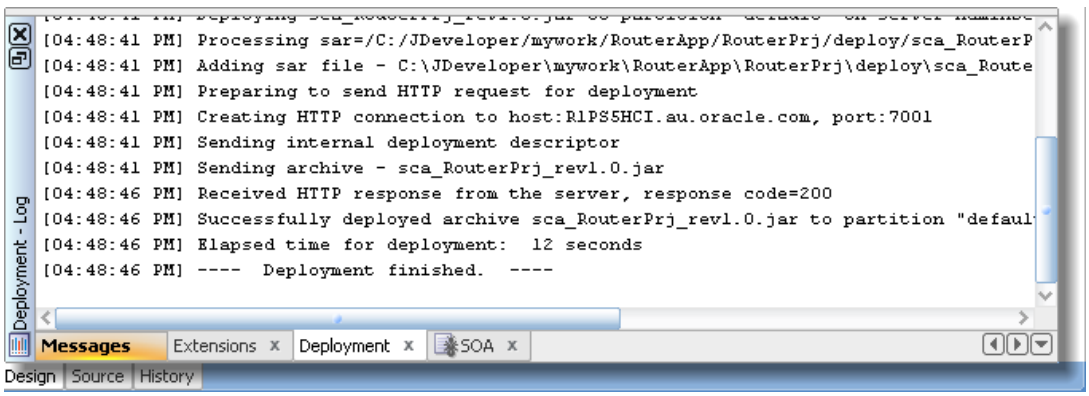
☐ Accept default "Deploy to Application Server" and click "Next"

☐ Check the "Overwrite any existing composite with the same revision ID." checkbox and click "Next"



☐ Choose the correct application server connection and click "Finish"

☐ Observe deployment log messages looking for completion without errors



The application is deployed and ready to accept and write messages.

At this point you can close the JDeveloper Studio IDE. We don't need it for the moment.

Send ADT messages

We will use the CMDHL7Sender command line client to read files containing a single HL7 ADT message and submit them to the ADT Receiver endpoint. We will then look at the console output produced by the CMDHL7Listener which we started earlier, then look at the output in our configured output directory – for me c:\hl7\received, and finally review message tracking information in the Healthcare Integration Console.

Please note that in this solution the receiver endpoint returns immediate ACK as soon as it gets the message. There may be a delay, most noticeable the first time one executes the processing flow after application server restart, between the receipt of the ACK and the time the message is written to a file in the file system.

☐ Check that your configured output directory is empty

☐ Locate the input file containing a single HL7 message - for me this will be C:\hl7\adt\sources\ADT_A01_output_1.hl7

The content of my file, where each segment starting with the 3 character segment ID in bold text is a single line up to the next 3 character segment ID, looks like this:

```
MSH|^~\&|SystemA|HosA|PI|MDM|2008090801529||ADT^A01|000000_CTLID_2008
090801529|P|2.3.1|||AL|NE
EVN|A01|2008090801529|||JavaCAPS6^^^^^^^USERS
PID|1||A000010^^^HosA^MR^HosA||Kessel^Abigail||19460101123045|M|||7
South 3rd Circle^^Downham Market^England -
Norfolk^30828^UK||||||||A2008090801529
PV1|1|I||I|||FUL^Fulde^Gordian^^^^^^^^^^MAIN|||EMR|||||||||V200809080
1529^^^^VISIT|||||||||||||||||||||||||2008090801529
```

☐ In a command / terminal window execute the following command

```
java -jar c:\tools\CMDHL7\CMDHL7Sender_v0.7.jar -a SystemA -b HosA -c ID_
-n 1 -d \r\r\n -p 22222 -h localhost -t 30000 -f
c:\hl7\adt\sources\ADT_A01_output_1.hl7
```

☐ Locate the output file in the received directory and inspect it to confirm that a) it has been written and b) that is has the same content as the input file

The content of my output file, where each segment starting with the 3 character segment ID in bold text is a single line up to the next 3 character segment ID, looks like this:

```
MSH|^~\&|SystemA|HosA|PI|MDM|2008090801529||ADT^A01|ID__0000000|P|2.3
.1|||AL|NE
EVN|A01|2008090801529|||JavaCAPS6^^^^^^^USERS
PID|1||A000010^^^HosA^MR^HosA||Kessel^Abigail||19460101123045|M|||7
South 3rd Circle^^Downham Market^England -
Norfolk^30828^UK||||||||A2008090801529
PV1|1|I||I|||FUL^Fulde^Gordian^^^^^^^^^^MAIN|||EMR|||||||||V200809080
1529^^^^VISIT|||||||||||||||||||||||||2008090801529
```

The content of the file is the same as the message which was sent. The only difference is the message control id, which the send command explicitly changed with the –c switch to a serial number prefixed by "ID__".

☐ Submit the ADT A03 file, `ADT_A03_output_1.hl7`, and inspect the output.

Our solution works to the extent of receiving HL7 v2.3.1 messages, and acknowledging them and writing them to files in the file system.

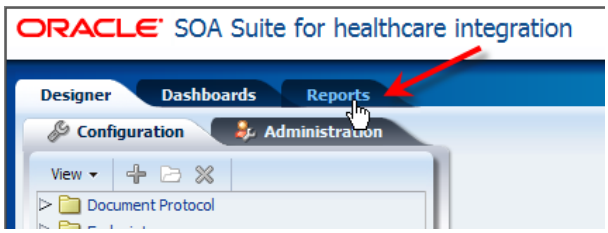
Explore Message Tracking

Let's explore message tracking.

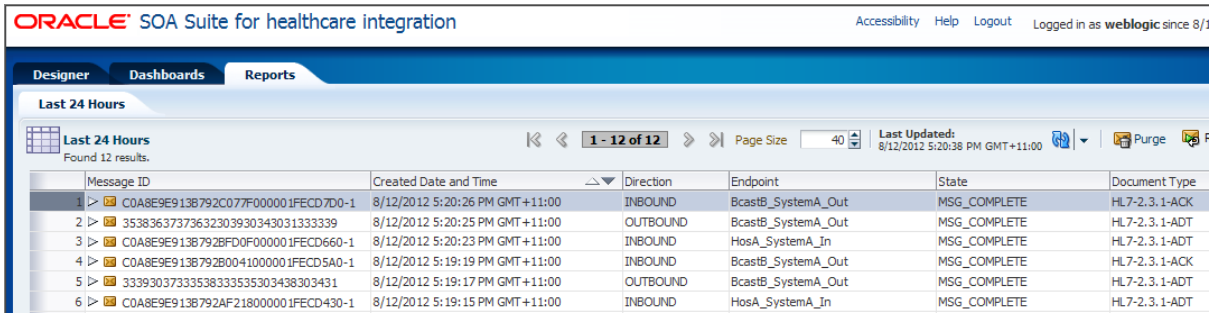
☐ Start the Healthcare Integration Console – http://localhost:7001/healthcare

☐ Log in with your credentials – mine are weblogic/welcome1

☐ Click the "Reports" Tab

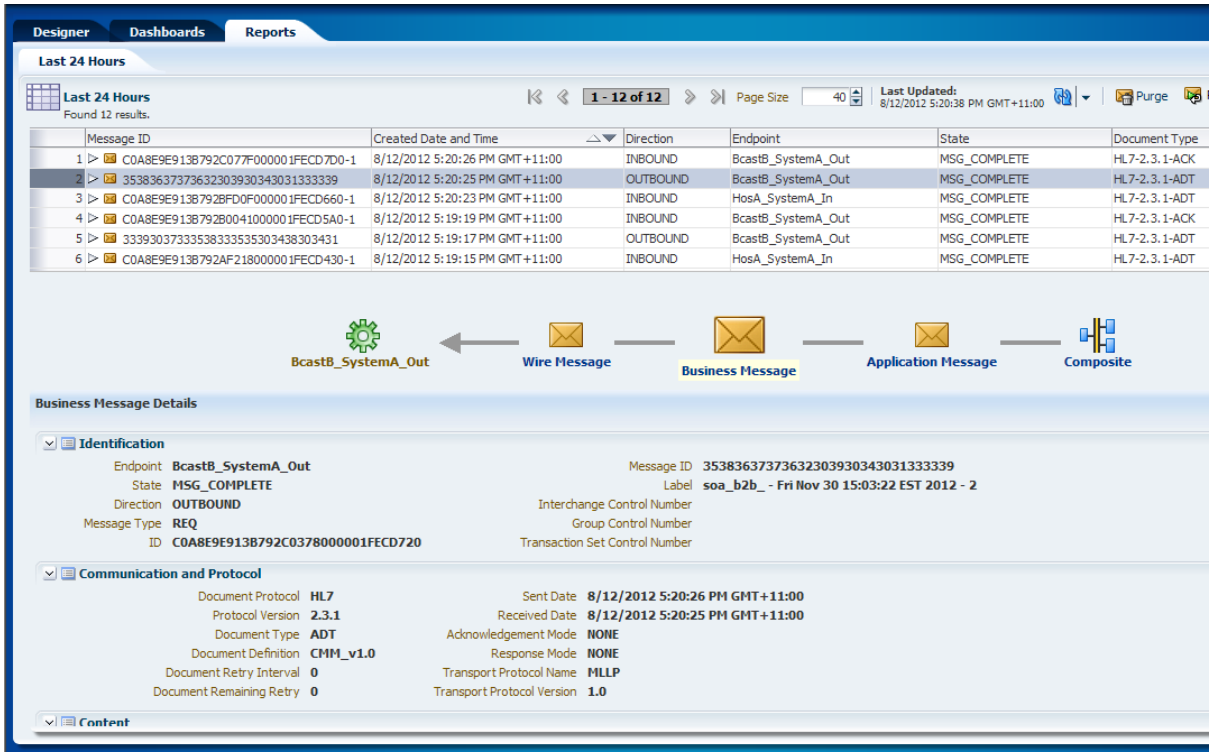


☐ Note one set of three messages for ADT A03 received (INBOUND) by the HosA_SystemA_In endpoint, one message sent (OUTBOUND) by BcastB_SystemA_Out endpoint and one message received (INBOUND) by BcastB_SystemA_Out endpoint, and one set of three messages for the ADT A01 message.



	Message ID	Created Date and Time	Direction	Endpoint	State	Document Type
1	C0A8E9E913B792C077F000001FECD7D0-1	8/12/2012 5:20:26 PM GMT+11:00	INBOUND	BcastB_SystemA_Out	MSG_COMPLETE	HL7-2.3.1-ACK
2	353836373736323039303430313333339	8/12/2012 5:20:25 PM GMT+11:00	OUTBOUND	BcastB_SystemA_Out	MSG_COMPLETE	HL7-2.3.1-ADT
3	C0A8E9E913B792BFD0F000001FECD660-1	8/12/2012 5:20:23 PM GMT+11:00	INBOUND	HosA_SystemA_In	MSG_COMPLETE	HL7-2.3.1-ADT
4	C0A8E9E913B792B0041000001FECD5A0-1	8/12/2012 5:19:19 PM GMT+11:00	INBOUND	BcastB_SystemA_Out	MSG_COMPLETE	HL7-2.3.1-ACK
5	333930373335383335353038303431	8/12/2012 5:19:17 PM GMT+11:00	OUTBOUND	BcastB_SystemA_Out	MSG_COMPLETE	HL7-2.3.1-ADT
6	C0A8E9E913B792AF218000001FECD430-1	8/12/2012 5:19:15 PM GMT+11:00	INBOUND	HosA_SystemA_In	MSG_COMPLETE	HL7-2.3.1-ADT

☐ Select the first OUTBOUND message in the list (it ought to be the outbound ADT A03 message)



☐ Note the details

☐ Explore message tracking for the message exchanges, noting inbound and outbound messages, payloads, properties and other details of interest

Note that both the Application Message and the Business Message payloads are HL7 v2 delimited messages. This is because we disabled HL7 v2 delimited to HL7 v2 XML translation at the time we configured this endpoint. Had we not done so, we would have seen XML messages in these cases.

Note that from now on whenever the expression "Inspect the Wire Message/Business Message/Application Message" or a similar expression is used it refers to the functionality just discussed as the means to perform this "inspection".

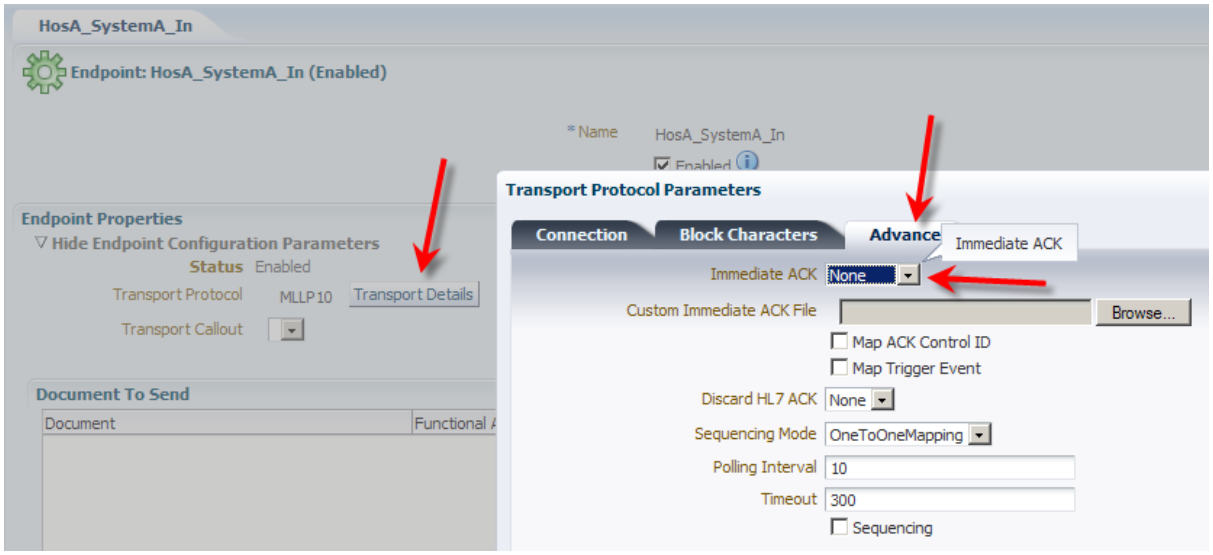
HL7 v2 Inbound to Outbound with XML translation

In the previous section we received a HL7 v2 delimited message and sent it out to an external systems without transformation. In this section we will re-configure the receiving endpoint to cause the message to be automatically translated to XML. We will also re-configure acknowledgement processing so that the Functional Acknowledgement is sent after the message is parsed and translated, and that it correctly indicates the outcome of translation and validation, if any.

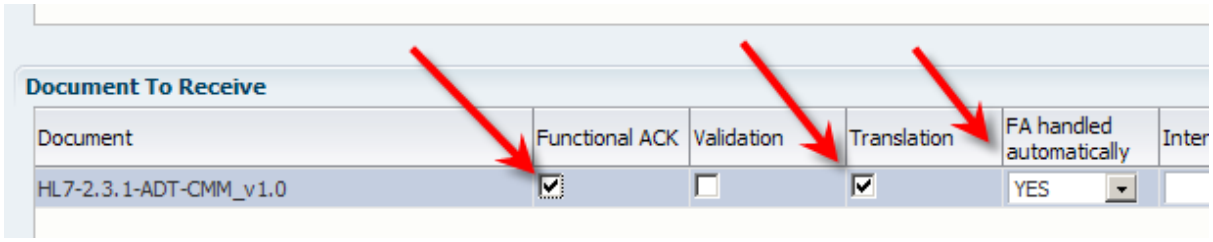
Configure Translation and Functional Acknowledgement in Inbound Endpoint

☐ In the Healthcare Integration Console expand the Designer→Configuration→Endpoint hierarchy and double-click the name of the endpoint "HosA_SystemA_In" to open the configurator pane

☐ Click the "Transport Details" button, select the "Advanced" Tab in the "Transport Protocol Parameters" dialogue, change "Immediate ACK" to ""none" and click "OK"



☐ In the "Document To Receive" section of the "HosA_SystemA_In" Tab check the "Functional ACK" checkbox, check the "Translation" checkbox, choose "YES" for the "FA handled automatically" dropdown then click "Apply" to apply changes

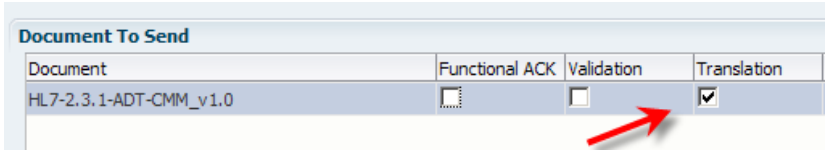


Configure Translation and Functional Acknowledgement in Outbound Endpoint

☐ In the Healthcare Integration Console expand the Designer→Configuration→Endpoint hierarchy and double-click the name of the endpoint "BcastB_SystemA_Out" to open the configurator pane

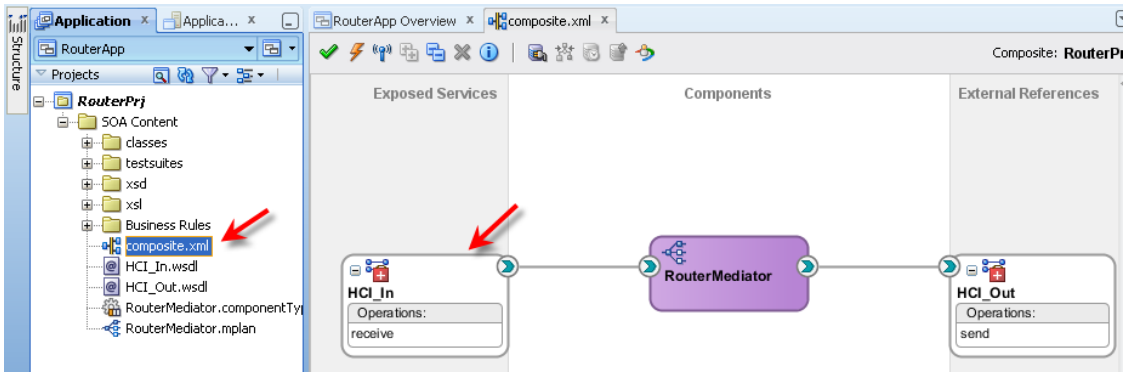
☐ Click the "Transport Details" button, navigate to "Advanced" Tab, set "Discard HL7 ACK" to "None" and click "

☐ In the "Document To Send" section of the "BcastB_SystemA_Out" Tab check the "Translation" checkbox then click "Apply" to apply changes

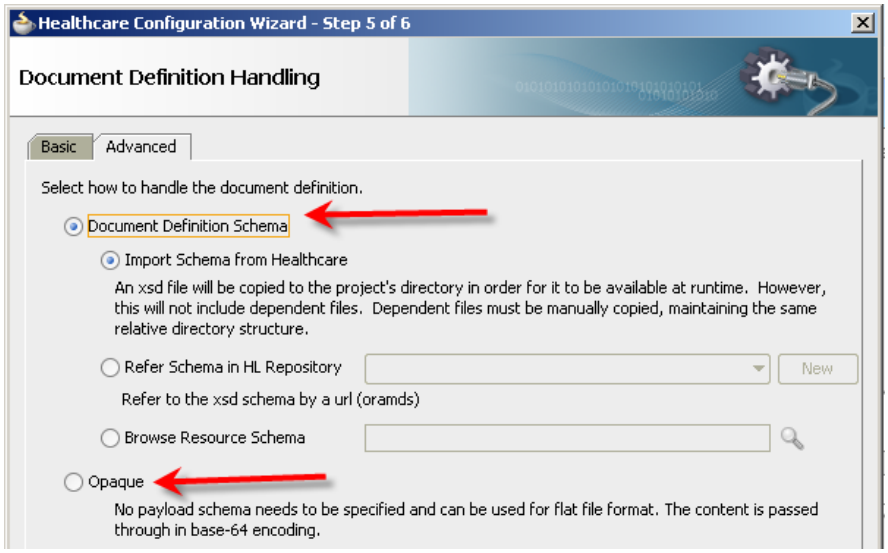


Re-configure and re-deploy the Router project

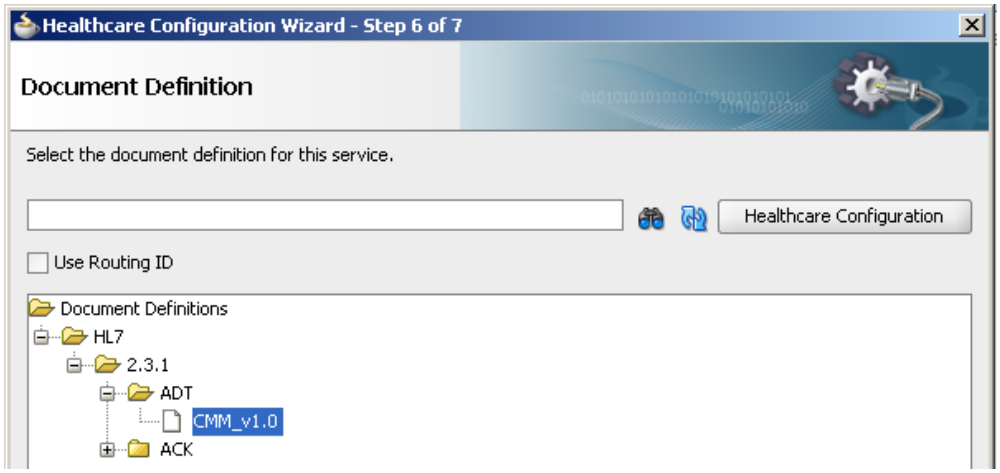
☐ Start JDeveloper Studio, if not already running, locate and double-click the composite.xml in the RouterApp's RouterPrj project to open the composite graphical editor, the double-click the HCI_ In adapter to start its configuration wizard



☐ Click "Next" until you get to the "Document Definition Handling" dialogue panel, click the "Document Definition Schema" to select it and deselect the "Opaque", which was the previous configuration option, then click "Next"



☐ Select the "CMM_v1.0" document from the "Document Definitions" hierarchy, then click "Next" and "Finish"



☐ Double-click the "HCI_Out" adapter to start its configuration wizard, click "Next" until you see the "Document Definition Handling" dialogue panel, click the "Document Definition Schema" to select it and deselect the "Opaque", which was the previous configuration option, then click "Next"

☐ Click "Next" and "Finish" to complete and dismiss the adapter configuration wizard

☐ Select the "CMM_v1.0" document from the "Document Definitions" hierarchy, then click "Next" and "Finish"

☐ Click the "Save All" button in JDeveloper toolbar, then deploy the project

☐ Close JDeveloper

Process ADT messages

We will again use the CMDHL7Sender command line client to read files containing single HL7 ADT messages and submit then to the ADT Receiver endpoint. We will then look at the output in our configured output directory – for me c:\hl7\received, look at the CMDHL7Listener console output and review message tracking information in the Healthcare Integration Console.

Please note that in this solution the receiver endpoint returns a Functional ACK as when it gets and parses the message.

☐ Check that your configured output directory is empty and delete any files it contains if it is not empty

☐ Locate the input file containing a single HL7 message - for me this will be C:\hl7\adt\sources\ADT_A01_output_1.hl7

The content of my file, where each segment starting with the 3 character segment ID in bold text is a single line up to the next 3 character segment ID, looks like this:

```
MSH|^~\&|SystemA|HosA|PI|MDM|2008090801529||ADT^A01|000000_CTLID_2008
090801529|P|2.3.1|||AL|NE
EVN|A01|2008090801529|||JavaCAPS6^^^^^^^USERS
PID|1||A000010^^^HosA^MR^HosA||Kessel^Abigail||19460101123045|M|||7
South 3rd Circle^^Downham Market^England -
Norfolk^30828^UK||||||||A2008090801529
PV1|1|I||I|||FUL^Fulde^Gordian^^^^^^^^^^MAIN|||EMR|||||||||V200809080
1529^^^^VISIT|||||||||||||||||||||||||2008090801529
```

☐ In a command / terminal window execute the following command

```
java -jar c:\tools\CMDHL7\CMDHL7Sender_v0.7.jar -a SystemA -b HosA -c ID_
-n 1 -d \r\r\n -p 22222 -h localhost -t 60000 -f
c:\hl7\adt\sources\ADT_A01_output_1.hl7
```

☐ Locate the output file in the received directory and inspect it to confirm that a) it has been written and b) that is has the same content as the input file

Part of the content of my output file (where I removed some of the content for brevity of display) looks like this:

```
<?xml version="1.0" encoding="UTF-8" ?><ADT
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" XDataVersion="2.0"
Standard="HL7" Version="2.3.1" CreatedDate="2012-11-11T14:54:33"
CreatedBy="XEngine_2992" GUID="{5EF84DB8-E35E-47B5-B134-EA02092AB49E}"
xmlns="http://www.edifecs.com/xdata/200">
   <Internal-Properties>
      <Data-Structure Name="Message">
         <Lookup Name="InternatCodeAlternateID"/>
         <Lookup Name="InternatCodeAlternateSystem"/>
         <Lookup Name="InternatCodeAlternateText"/>
...
         <Lookup Name="MessageVersion">2.3.1</Lookup>
```

```
         <Lookup Name="Standard">HL7</Lookup>
         <Lookup Name="TriggerEvent">A01</Lookup>
         <Property Name="AcceptAckType">AL</Property>
         <Property Name="AlternateCharacterSetSchema"/>
         <Property Name="AppAckType">NE</Property>
...
         <Property Name="Standard">HL7</Property>
         <Property Name="SubcomponentDelimiter">0x26</Property>
         <Property Name="SubelementDelimiter">0x5e</Property>
         <Property Name="TriggerEvent">A01</Property>
      </Data-Structure>
   </Internal-Properties>
   <MSH>
      <MSH.1>|</MSH.1>
      <MSH.2>^~\&amp;</MSH.2>
      <MSH.3>
         <HD.1>SystemA</HD.1>
      </MSH.3>
      <MSH.4>
         <HD.1>HosA</HD.1>
      </MSH.4>
      <MSH.5>
         <HD.1>PI</HD.1>
      </MSH.5>
      <MSH.6>
         <HD.1>MDM</HD.1>
      </MSH.6>
      <MSH.7>2008090801529</MSH.7>
```

The content of the file is the XML "equivalent" of the message which was sent. Note the "Internal-Properties" XML structure. The property values are derived from the message structure, messaging environment (endpoint and document configuration) and message content itself, and are carried with the message to the SOA Composite, where they can be accessed and used as might be required.

☐ Submit the ADT A03 file, `ADT_A03_output_1.hl7`, and inspect the output.

Our solution works to the extent of receiving HL7 v2.3.1 messages, acknowledging them, passing them to the outbound endpoint which sends them to an external system, which write them to files in the file system.

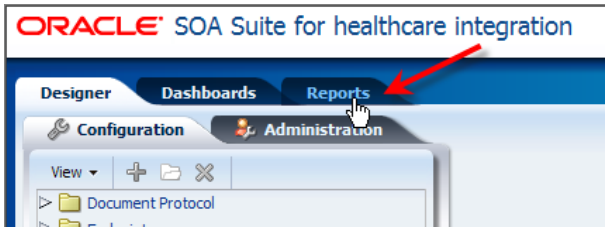
Explore Message Tracking

Let's explore message tracking.

☐ Start the Healthcare Integration Console – http://localhost:7001/healthcare

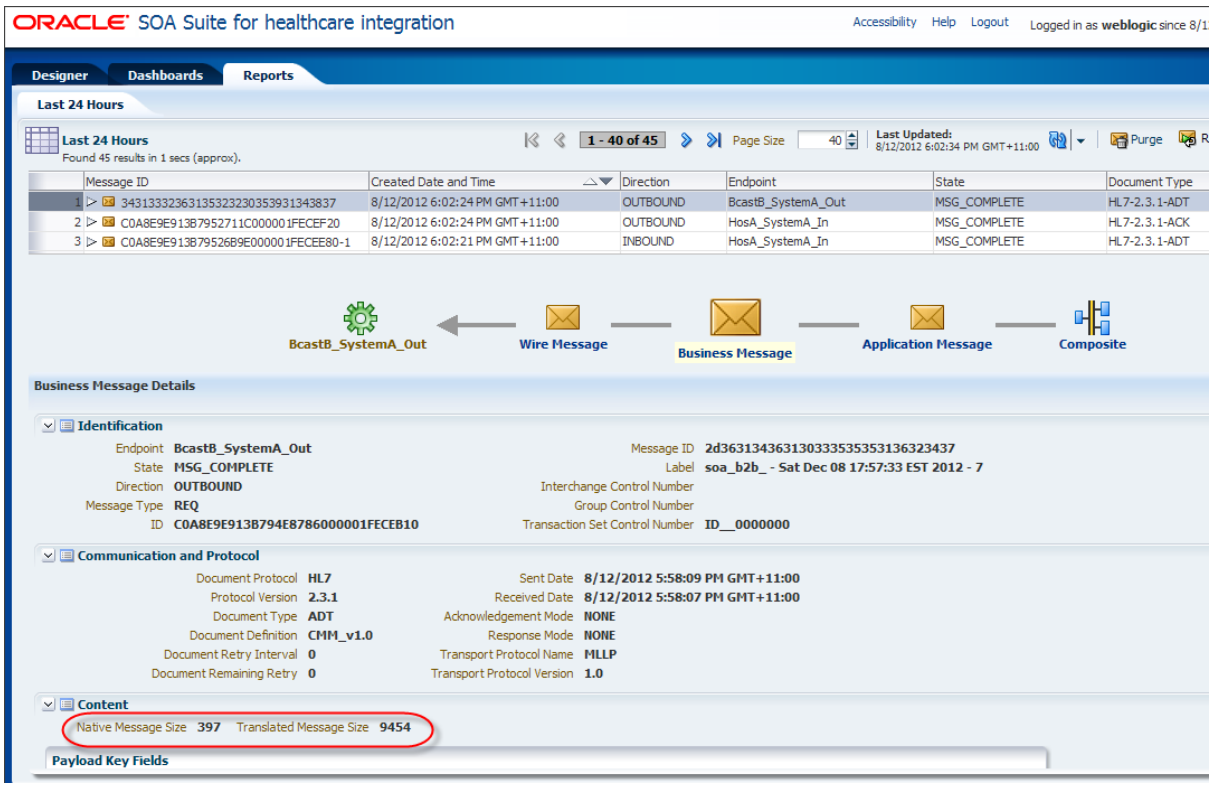
☐ Log in with your credentials – mine are weblogic/welcome1

☐ Click the "Reports" Tab

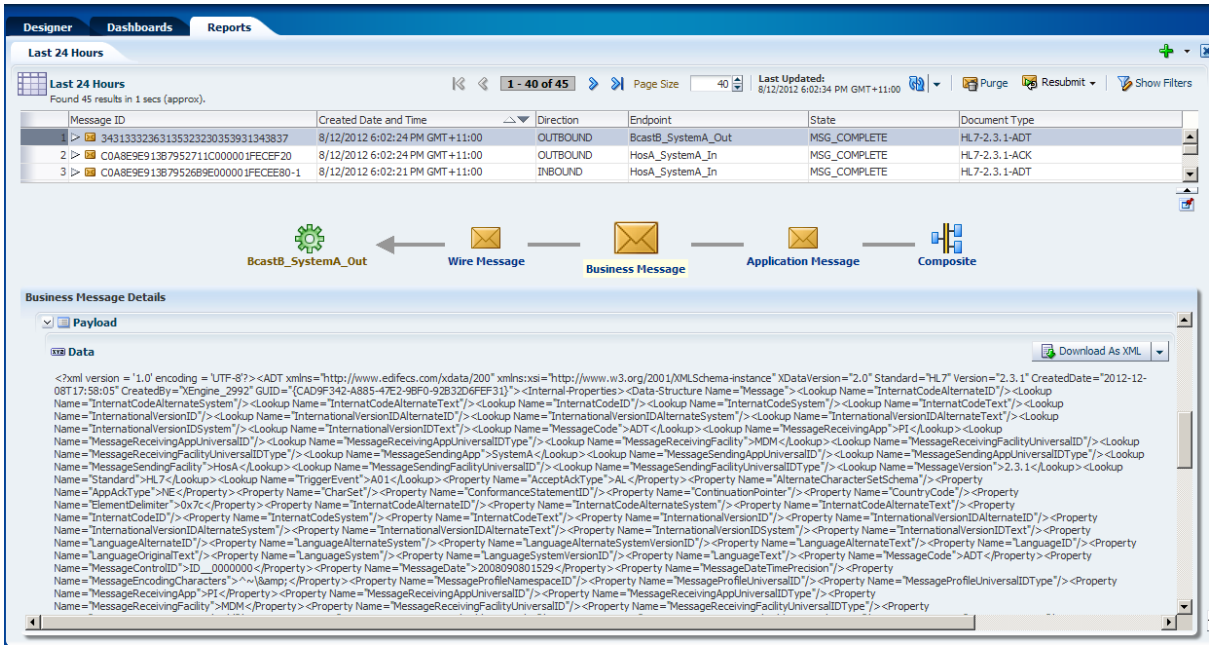


☐ Select the "OUTBOUND" message for the : BcastB_SystemA_Out" endpoint, review the State, Endpoint and other attributes, and review "Identification" and "Communication and Protocol" attribute sections – you should recall most of these from the endpoint

configuration steps – note the "Native Message Size" and "Translated Message Size" information



☐ Scroll down the "Business Message Details" pane, expand the "Payload" node and inspect the HL7 message payload, noting the "Download as XML" button, which allows the payload to be "externalized", i.e. saved into a file in the file system – the "Business Message" is the translated message in the XML format

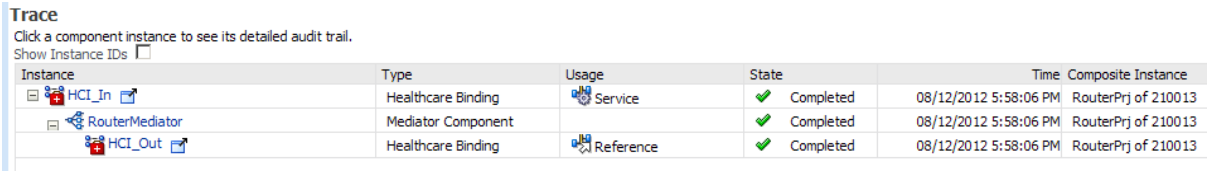


☐ Click the "Wire Message" Icon or link, inspect the wire message-related attribute values and the message content – the "Wire Message" payload is the original HL7 v2 Delimited format

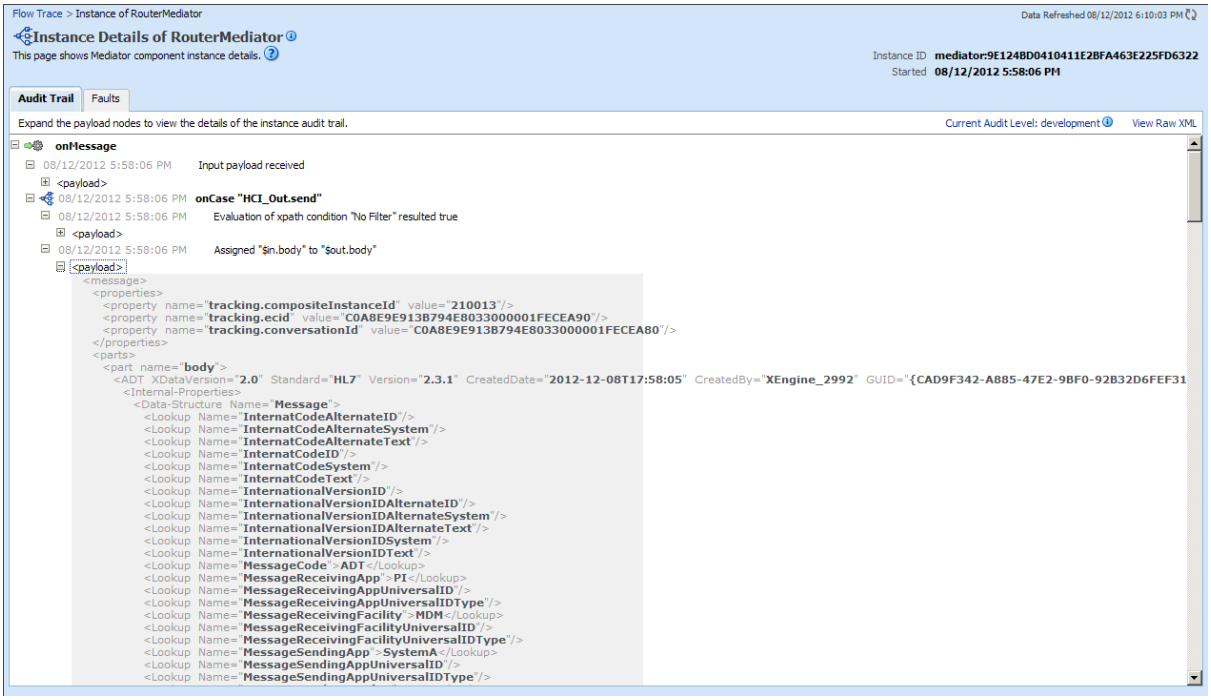
☐ Click the "Application Message" icon/link, inspect attribute values (paying particular attention to "Key" and "Value" columns in "Message Properties") , expand the "Payload" node and inspect the HL7 message

Note that both the Application Message and the Business Message payloads are HL7 v2 XML messages. This is because we enabled HL7 v2 delimited to HL7 v2 XML translation at the time we configured this endpoint.

☐ Click the "Composite" link, log into the Enterprise Manager Console with your credential (my credentials are weblogic/welcome1) and review the message processing "Trace", noting component hierarchy, component names, types, state and so on



☐ Click the "RouterMediator" link and inspect the instance details, expanding nodes as you go along to see what can be seen – this display shows the SOA Composite and the message and message properties as they are at different stages of processing



We are done with the inbound HL7 v2 delimited to outbound HL7 v2 delimited example. Future examples will explore more routing and transformation.

Summary

This article worked through the mechanics of configuring the "SOA Suite for healthcare integration" to receive a HL7 v2.3.1 ADT message as a Canonical Message, transforming it to XML, routing it to an outbound endpoint, transforming it back to HL7 v2 delimited format and sending it to an externa;l system - an expanded "Hello World" solution in a HL7 messaging environment.

Both pass-through with no translation and delimited to XML translation solutions were implemented, extending functionality to use messaging environment and message content properties in naming files.