Oracle SOA Suite 11g R1 PS5

SOA Suite for healthcare integration Series

HL7 v2 solution using JMS "the Java CAPS way"

michael@czapski.id.au

February 2013

Table of Contents

Introduction

This article may be of interest to these who would like to use the "SOA Suite for healthcare integration" HL7 v2 delimited message handling functionality in solutions similarly to how Oracle Java CAPS HL7 eWay-based solutions were built, perhaps as endpoints in a "Service Bus"-based infrastructure, or to these who would like to use the HL7 messaging handing functionality in OSB environments. In essence, for these unfamiliar with the Java CAPS pattern of use, there were the "Inbound HL7 eWay" and the "Outbound HL7 eWay" patterns. An inbound HL7 v2 Adapter (eWay) would receive a HL7 message, perform all (minimal) validation and acknowledgement processing and store the incoming message in a persistent JMS Queue for some downstream component to process the message as necessary. An Outbound HL7 v2 Adapter (eWay) would read a HL7 message from a JMS Queue (where it was deposited by some upstream component) and send it out to the external system, performing any HL7 ACK processing that might have been required.

In the prior articles in this series we used direct integration between the HL7 v2 endpoints and SOA Suite Composites which provided processing logic. While in my articles JMS is used implicitly as an internal mechanism (via B2B_IN_QUEUE and B2B_OUT_QUEUE JMS queues) the SOA Suite composite did not explicitly use JMS adapters.

It is possible to configure SOA Suite for healthcare integration endpoints in such a way that messages the inbound endpoint receives will be deposited in a particular JMS destination (queue or topic other than the B2B_IN_QUEUE) associated with the endpoint, and messages to be sent by an outbound endpoint will originate in a specific JMS destination (queue or topic other than the B2B_OUT_QUEUE). This will allow such endpoints to be used in Oracle Service Bus-based or other ESB-based solutions as services with JMS interfaces.

In this article we will develop and exercise an inbound-to-JMS and JMS-to-outbound HL7 v2 delimited message processing solutions to demonstrate this capability.

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.

Java CAPS Note

A HL7 processing solution in Java CAPS will typically receive HL7 v2 Delimited messages through the HL7 eWay, transform them in some way, and potentially send them on to HL7 receivers through a HL7 eWay. Standard HL7 processing, acknowledgements, message header validation, sequence number processing, are handed by pre-built Java CAPS projects, which are available as part of the installation and must be imported and potentially modified for use in a solution. The inbound project, prjHL7Inbound, receives HL7 messages and deposits them in a JMS Queue for a downstream solution to process further. The outbound project, prjHL7Outbound, receives HL7 messages from a JMS Queue and sends them on to the receivers. The site-specific transformations and message processing happens in one or more components, the initial of which receives messages from the JMS Queue to which the HL7 Inbound sent them, and the final of which ultimately deposits messages in a JMS Queue for the HL7 outbound to send. The complexity involved in transformational of messages, access to various enterprise resources and message manipulation will vary from solution to solution.

The schematic below shows major components involved in a typical Java CAPS HL7 solution described above.

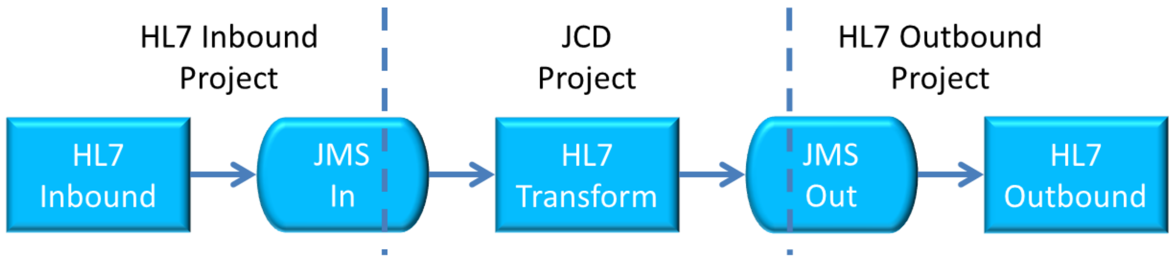


Figure 1, Simplest possible Java CAPS JCD JL7 Solution

If we construct a Java CAPS solution in such a way that the HL7 Inbound, the HL7 Transform and the HL7 Outbound are implemented as separate Java CAPS projects we might get a project hierarchy like the one shown below.

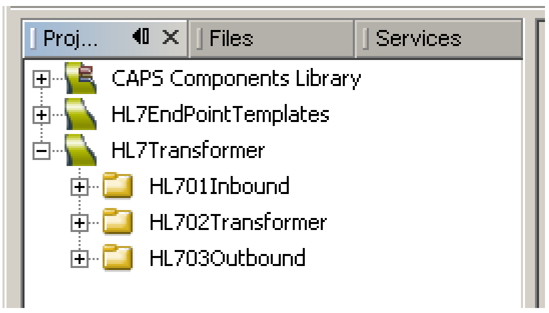


Figure 2, HL7 Transformer Project Hierarchy

The HL7 Inbound Connectivity Map for the HL701Inbound, which is derived from prjHL7Imbound, is shown below.

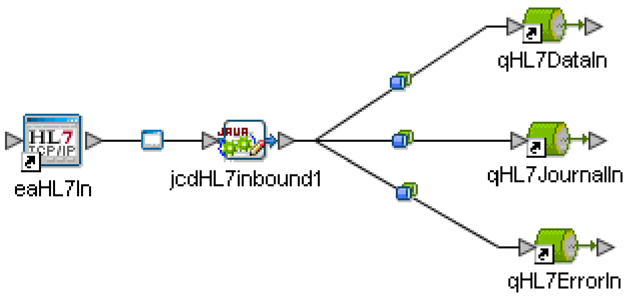


Figure 4, HL7 Inbound Connectivity Map

The HL7 eWay receives messages and deposits them in the JMS Queue called qHL7DataIn. The JCD itself is the unmodified JCD imported with the project prjHL7Inbound. It handles all HL7-related communication functionality including ACKs.

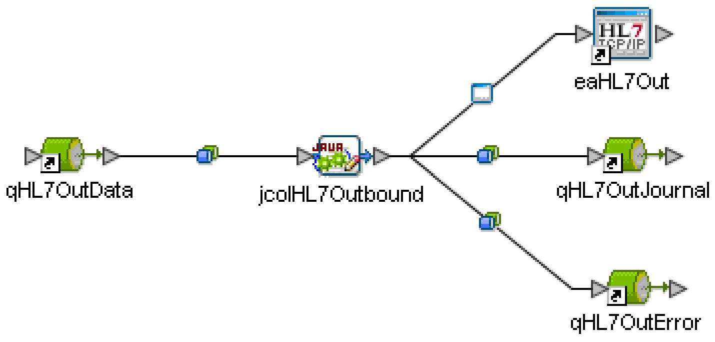


Figure 3, HL7 Outbound Connectivity Map

The Connectivity Map for the HL703Outbound is shown below.

The HL7 eWay sends messages it reads from the JMS Queue called qHL7OutData. The JCD itself is the unmodified JCD imported with the project prjHL7Outbound. It handles all HL7-related communication functionality including ACKs.

The connectivity map for the HL702Transformer project is simplicity itself and requires no elaboration.

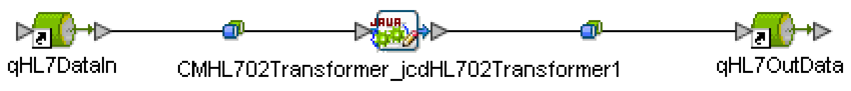


Figure 5, HL7 Transformer Connectivity Map

The collaboration receives a message from qHL7DataIn, transforms it in some manner and deposits the resulting message in qHL7OutData.

To anticipate what will follow let's say upfront that the HL701Inbound and the HL703Outbound projects will be replaced, in their entirety, by the "Oracle SOA Suite for healthcare integration" infrastructure with correctly configured Endpoints, therefore there will be no further discussion of these projects. The HL702Transformer project will be re-implemented using a SOA Suite Mediator Component in a Composite. Other forms of logic components could be used, for example logic hosted in the Oracle Enterprise Service Bus, as long as they support reading from/writing to JMS Queues hosted in an Oracle WebLogic Server.

Solution Overview

The point of developing the solution set discussed in this article is to allow a SOA Suite-based HL7 v2 delimited message processing solution to be architected as a SEDA (Staged Event-Driven Architecture) solution using JMS destinations as staging points. This is very much the way that Java CAPS HL7 v2 delimited processing solutions were typically architected.

We will replace the HL7 eWay and its associated JCD and JMS destinations with the "SOA Suite for healthcare integration" endpoints and use internal channels to direct HL7 messages to/from JMS queues.

We can replace the "HL7 Transform" component with any piece of logic which can receive messages from a JMS destination and send messages to a JMS destination. In this article a SOA Suite Mediator component is used but it could equally well be a piece of Oracle Service Bus-based logic or indeed any other piece of logic which can interact with WebLogic JMS destinations.

The overall architecture is depicted in Figure 6.

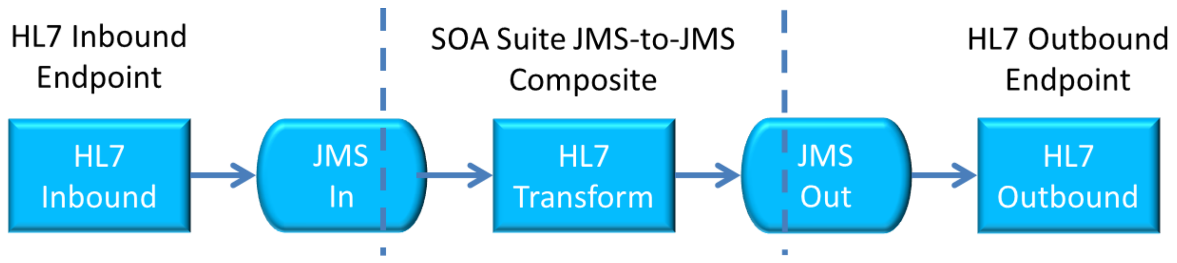


Figure 6, "SOA Suite for healthcare integration" HL7 Endpoint-based solution schematic

In this article the inbound HL7 v2 A01 messages will be received by the inbound endpoint and will be routed by the outbound endpoint to a receiver system. Message streams will be passed on unchanged.

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 adapters will translate messages 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 7.

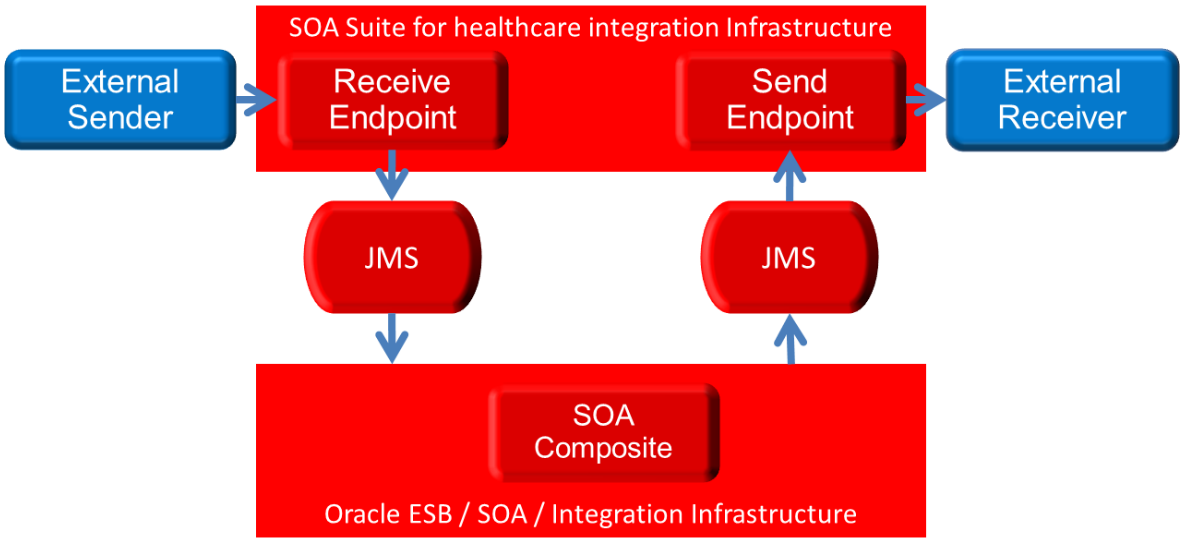

SOA Suite for healthcare integration Infrastructure
External Sender
Receive Endpoint
Send Endpoint
External Receiver
JMS
JMS
SOA Composite
Oracle ESB / SOA / Integration Infrastructure

Figure 7, Runtime components

To summarize:

An external system will send HL7 v2 delimited messages to the "SOA Suite for healthcare integration" endpoint, which will acknowledge them and deposit them in a JMS Queue.

A SOA Suite composite will use a mediator component to read messages from the JMS Queue construct required JMS user-defined properties and send them to the outbound JMS Queue.

A "SO Suite for healthcare integration" endpoint will be handed messages from a JMS Queue. It will translate them if appropriate and will send them to the external system.

The solution components are depicted in **Error! Reference source not found.**.

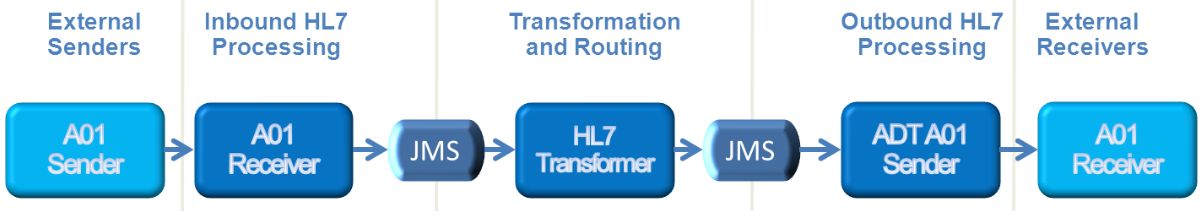


Figure 8, Solution Components

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.

A01 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 ADT A01 messages and will receive and display acknowledgements.

A01 Receiver is the CMDHL7 sender tool, or another tool capable of receiving HL7 v2 Delimited messages over TCP/IP using the MLLP protocol. It will receive ADT A01 messages and will send acknowledgements.

We will reuse CMM_v1.0 message structures / documents, configure inbound and outbound Endpoints and implement a forwarder composite application. This we will do in the subsequent sections.

Preliminaries – Queue Browser Tool

We will be working explicitly with JMS Queues. To make it easy to see what is going on, as well as to be able to explicitly delete messages from a queue or send messages to a queue, we will use a QBrowser tool. Please read the article I published about 1 ½ years ago, "Using QBrowser v2 with WebLogic JMS for 10.3", at http://blogs.czapski.id.au/2011/05/using-qbrowser-v2-with-weblogic-jms-for-10-3, for details of obtaining and configuring QBrowser for use with the WebLogic JMS infrastructure.

Preliminaries – Create JMS Queues in WebLogic JMS

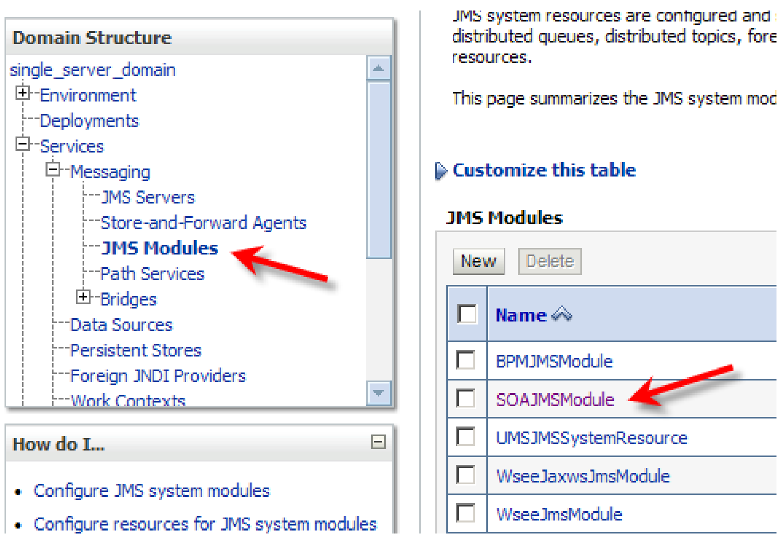
Unlike the Java CAPS, which would create JMS Queues and Topics in the supported STCMS or Sun Java MQ JMS servers on reference, the queues and topics hosted by the WebLogic JMS must be explicitly created. In this section we will create the connection factories and queues which will be used by the internal delivery channels associated with the inbound and outbound endpoints for passing messages to and from the HL7 Transformer component.

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.

To keep things simple we will do the minimum necessary to create two JMS queues, qHL7In and qHL7Out. To that end we will re-use the existing JMS server "SOAJMSServer" and the existing JMS Module ""SOAJMSModule.

- ☐ Start the WebLogic Administration Console, perhaps http://localhost:7001/console, and log in with your credentials, perhaps weblogic/welcome1

- ☐ Expand the "single_server_domain" node (or whatever yoyr WebLogic domain is called) in the "Domain Structure" pane at the right of the console window, "Services" → "Messaging", click "JMS Modules" and then click "SOAJMSModue" in the right hand pane



Domain Structure

single_server_domain
- Environment
- Deployments
- Services
  - Messaging
    - JMS Servers
    - Store-and-Forward Agents
    - JMS Modules
    - Path Services
    - Bridges
  - Data Sources
  - Persistent Stores
  - Foreign JNDI Providers
  - Work Contexts

How do I...

- Configure JMS system modules
- Configure resources for JMS system modules

JMS system resources are configured and
distributed queues, distributed topics, fore
resources.

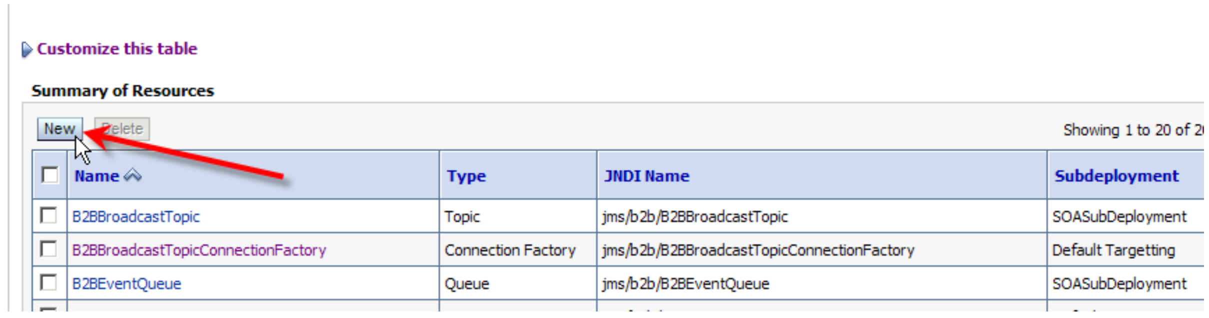
This page summarizes the JMS system moc

Customize this table

JMS Modules

Name
BPMJMSModule
SOAJMSModule
UMSJMSSystemResource
WseeJaxwsJmsModule
WseeJmsModule

☐ Click “New” under the “Summary of Resources”



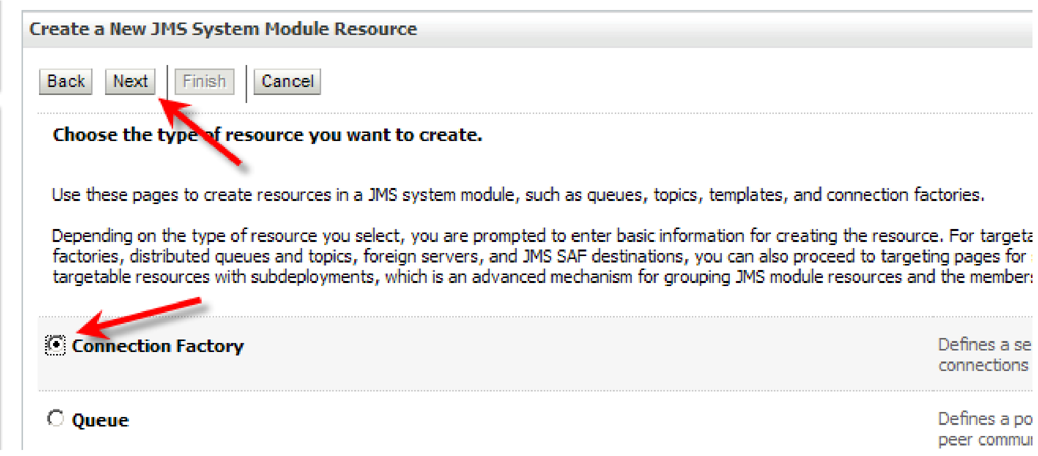
Customize this table

Summary of Resources

Showing 1 to 20 of 2

Name	Type	JNDI Name	Subdeployment
B2BBroadcastTopic	Topic	jms/b2b/B2BBroadcastTopic	SOASubDeployment
B2BBroadcastTopicConnectionFactory	Connection Factory	jms/b2b/B2BBroadcastTopicConnectionFactory	Default Targetting
B2BEventQueue	Queue	jms/b2b/B2BEventQueue	SOASubDeployment

☐ Select the “Connection Factory” radio button and click “Next”



Create a New JMS System Module Resource

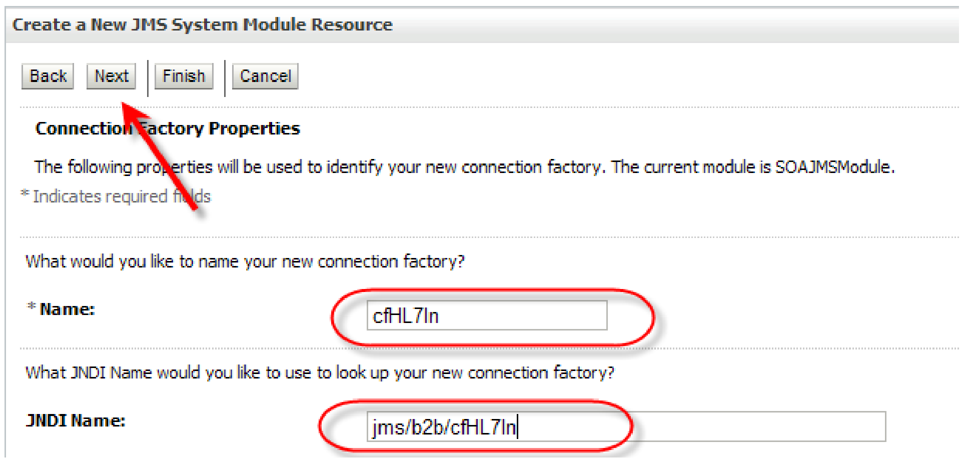
Choose the type of resource you want to create.

Use these pages to create resources in a JMS system module, such as queues, topics, templates, and connection factories.

Depending on the type of resource you select, you are prompted to enter basic information for creating the resource. For targeta
factories, distributed queues and topics, foreign servers, and JMS SAF destinations, you can also proceed to targeting pages for
targetable resources with subdeployments, which is an advanced mechanism for grouping JMS module resources and the member:

[✓ Connection Factory]	Defines a se connections
Queue	Defines a po peer commu

☐ Specify “cfHL7In” as name of the connection factory and “jms/b2b/cfHL7In” and JNDI name then click “Next”


Create a New JMS System Module Resource

Connection Factory Properties

The following properties will be used to identify your new connection factory. The current module is SOAJMSModule.

* Indicates required fields

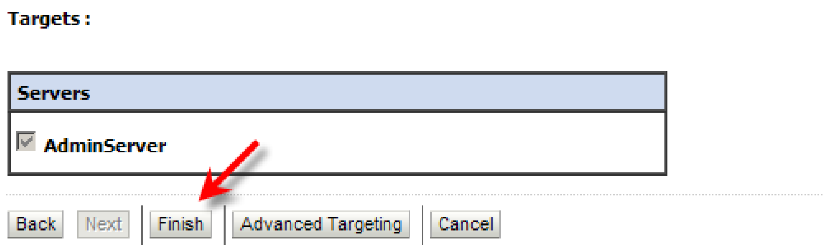
What would you like to name your new connection factory?

* Name: cfHL7In

What JNDI Name would you like to use to look up your new connection factory?

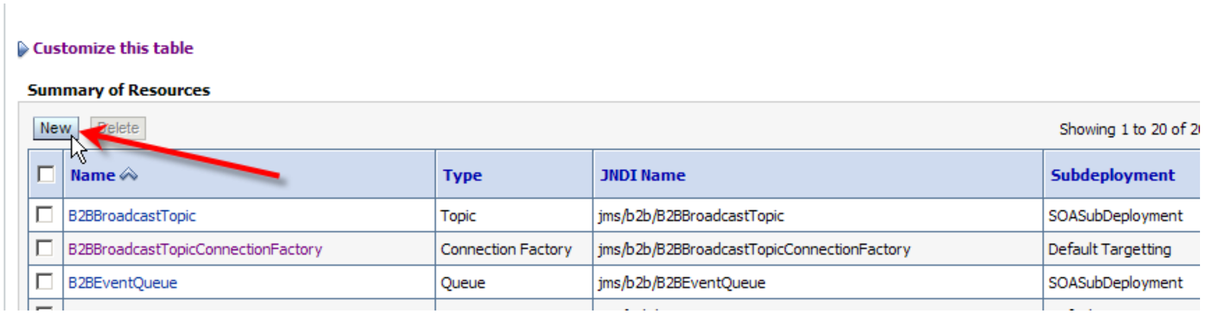
JNDI Name: jms/b2b/cfHL7In

- ☐ Choose the target server in your domain or accept the onlt server if you are using a development environment with no managed servers, as I do for these articles, the click "Finish" to complete creation of the connection factory belonging to the "SOAJMSModule" hosted in the "SOAJMSServer"


Targets :

Servers
☑ AdminServer

- ☐ Click "New" under the "Summary of Resources"

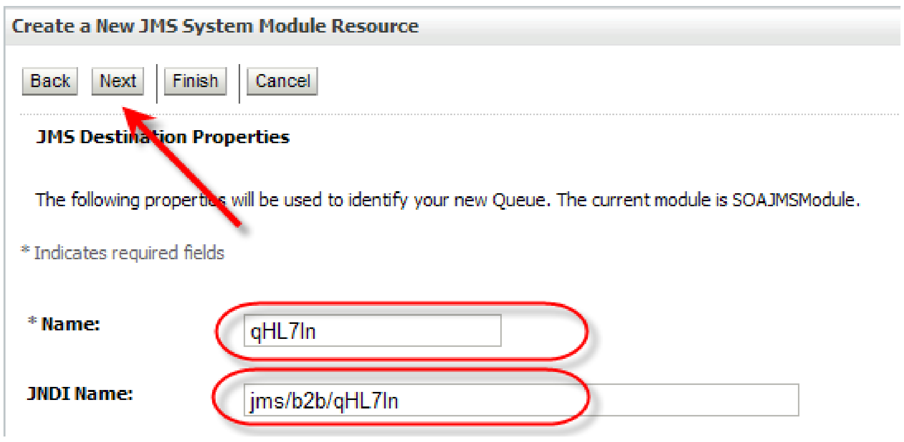


Summary of Resources

Showing 1 to 20 of 2

	Name	Type	JNDI Name	Subdeployment
☐	B2BBroadcastTopic	Topic	jms/b2b/B2BBroadcastTopic	SOASubDeployment
☐	B2BBroadcastTopicConnectionFactory	Connection Factory	jms/b2b/B2BBroadcastTopicConnectionFactory	Default Targetting
☐	B2BEventQueue	Queue	jms/b2b/B2BEventQueue	SOASubDeployment

- ☐ Select the "Queue" radio button and click "Next"
- ☐ Name the queue "qHL7In", give it the JNDI name of "jms/b2b/qHL7In" and click "Next"



Create a New JMS System Module Resource

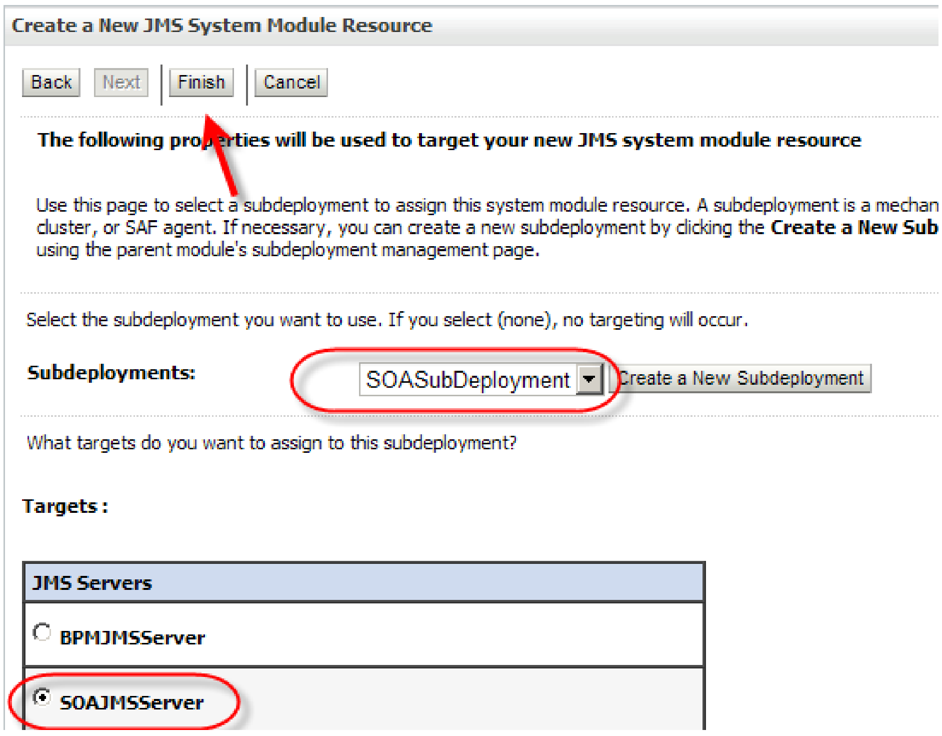
JMS Destination Properties

The following properties will be used to identify your new Queue. The current module is SOAJMSModule.

* Indicates required fields

* Name:	qHL7In
JNDI Name:	jms/b2b/qHL7In

☐ Choose "SOASubDeployment" from the "Subdeployments" drop-down, make sure the "SOAJMSServer" is selected and click "Finish" to complete creation of the JMS queue



Create a New JMS System Module Resource

The following properties will be used to target your new JMS system module resource

Use this page to select a subdeployment to assign this system module resource. A subdeployment is a mechan cluster, or SAF agent. If necessary, you can create a new subdeployment by clicking the **Create a New Sub** using the parent module's subdeployment management page.

Select the subdeployment you want to use. If you select (none), no targeting will occur.

Subdeployments: SOASubDeployment

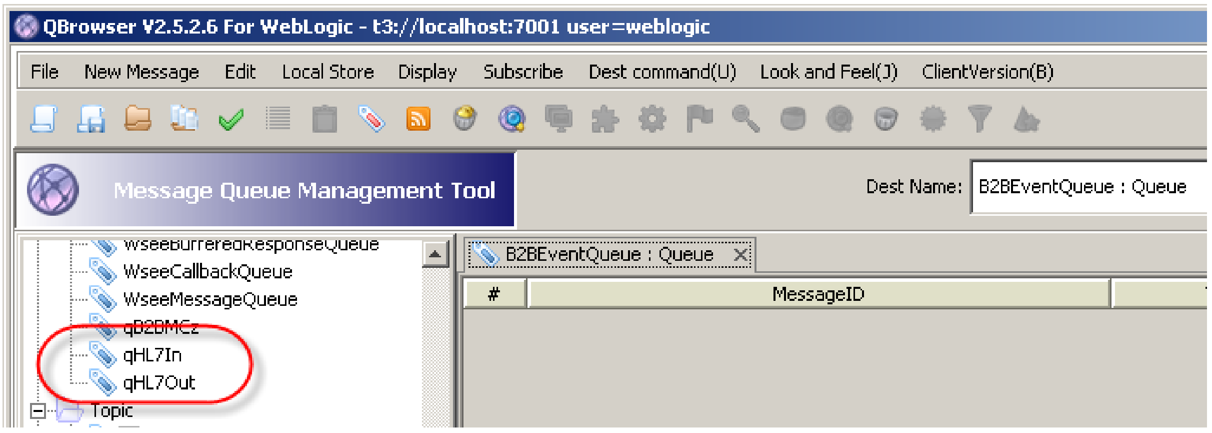
What targets do you want to assign to this subdeployment?

Targets :

JMS Servers
○ BPMJMSServer
● SOAJMSServer

☐ Repeat the steps to create "cfHL7Out" / "jms/b2b/cfHL7Out" connection factory and "qHL7Out"   "jms/b2b/qHL7Out" queue

☐ Use QBrowser to view the two new queues



Preliminaries – CMM_v1.0 and ACK Documents

It is assumed that the ADT messages will be cast from 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.

The CMM_v1.0 document must be available and "introduced" to the "SOA Suite for healthcare integration", as discussed in section "Add CMM_v1.0 Document to Document Protocol Hierarchy", page 4 and subsequent, in the article "SOA Suite for healthcare integration Series - HL7 v2 Inbound to File Solution" at http://blogs.czapski.id.au/2012/11/soa-suite-for-healthcare-integration-series-hl7-v2-inbound-to-file-solution.

It is assumed that the ADT ACK message will be received from the external system and will be discarded. To support this we need to have the ADT ACK document defined, as discussed in section "Define Functional Acknowledgement Document", page 27, and "introduced" to the "SOA Suite for healthcare integration", as discussed in section ""Introduce" the ACK document to SOA Suite for healthcare integration", pages 28-29 in the article "SOA Suite for healthcare integration Series - HL7 v2 Inbound to File Solution" at http://blogs.czapski.id.au/2012/11/soa-suite-for-healthcare-integration-series-hl7-v2-inbound-to-file-solution.

Preliminaries – Exception Handling

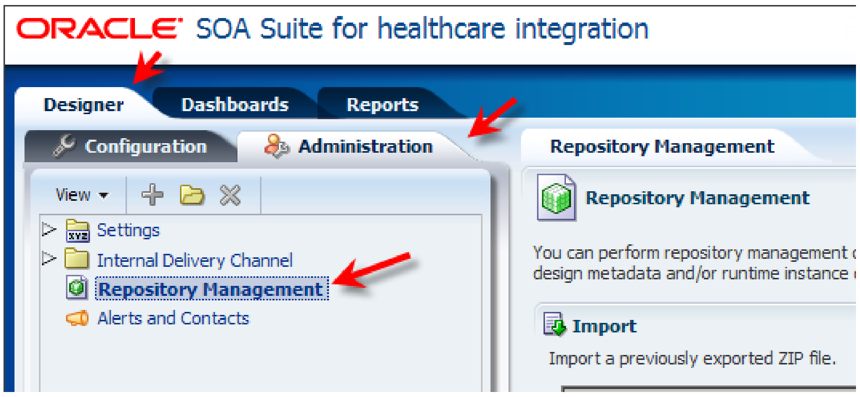
If you followed this series of articles and implemented solutions discussed in them you will have an exception handling solution in place. This solution will pick up exception messages from the B2B_IN_QUEUE and will write them to files in the file system. If you did not follow the series but would like to have this facility then implement the solution discussed in the article "SOA Suite for healthcare integration Series – Exception Handling – Processing Endpoint Errors" at http://blogs.czapski.id.au/2013/01/soa-suite-for-healthcare-integration-series-exception-handling-processing-endpoint-errors.

Clear Instances from the Repository

To start with a "clean slate" we will use the Healthcare Integration Console to clear out all old message tracking information. Bear in mind that this is irreversible and one should think carefully about clearing instance repository in production systems.

- ☐ Start the Healthcare Integration Console, http://localhost:7001/healthcare, and log in with your credentials, perhaps weblogic/welcome1

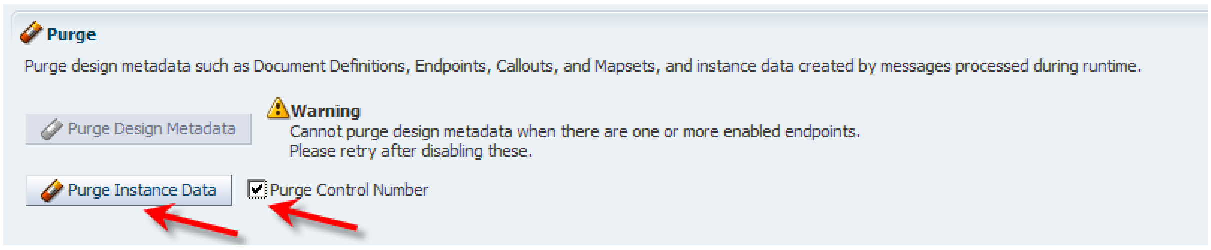
- ☐ Click "Designer" Tab, click "Administration" Tab and double-click on the "Repository Management" node



You can perform repository management design metadata and/or runtime instance

Import a previously exported ZIP file.

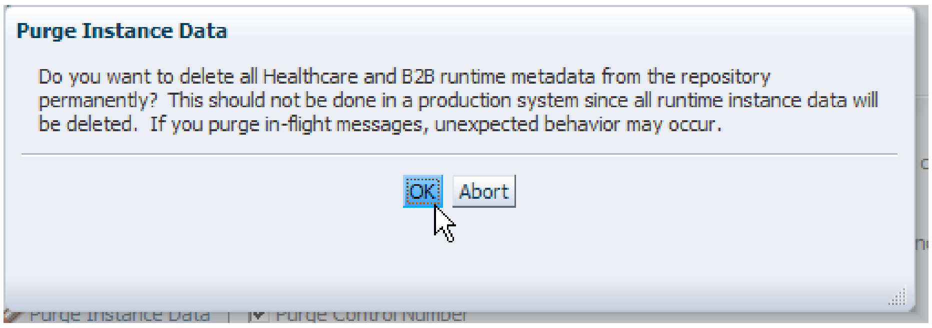
- ☐ Check the "Purge Control Number" checkbox, if appropriate, and click the "Purge Instance Data" button



Purge design metadata such as Document Definitions, Endpoints, Callouts, and Mapsets, and instance data created by messages processed during runtime.

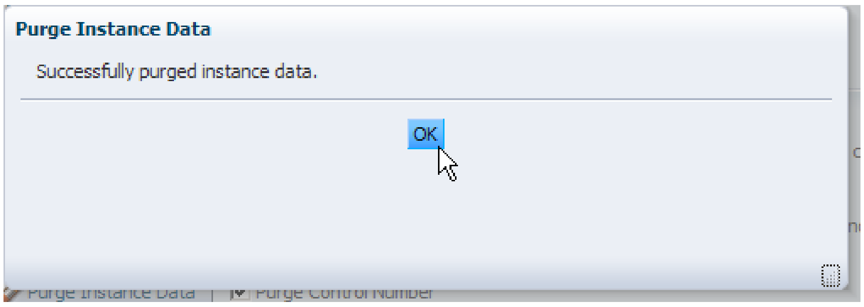
Cannot purge design metadata when there are one or more enabled endpoints. Please retry after disabling these.

- ☐ Click "OK" to purge messages and message tracking information from the repository



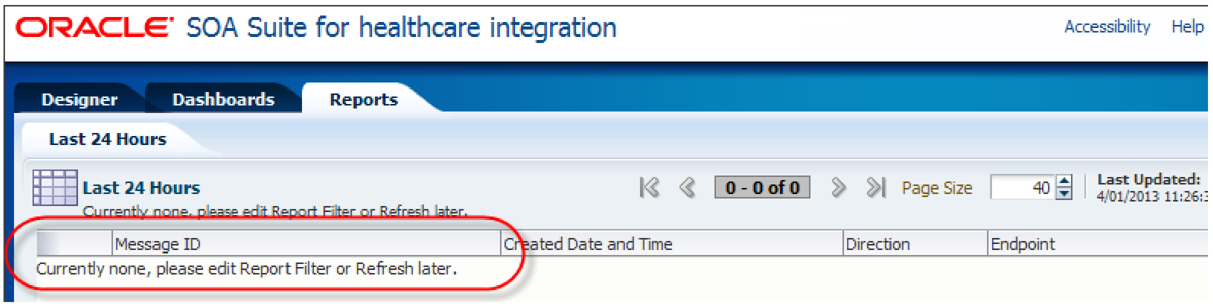
Do you want to delete all Healthcare and B2B runtime metadata from the repository permanently? This should not be done in a production system since all runtime instance data will be deleted. If you purge in-flight messages, unexpected behavior may occur.

- ☐ Once the process is completed, dismiss the wizard by clicking "OK"



Successfully purged instance data.

- ☐ Click on the "Reports" Tab, look at the last 24-hours report and note that there are no messages in the repository



Configure HL7JMSOut Endpoint

For greater educational value we will develop the solution a bit at a time, starting with the endpoint which delivers the payload from a JMS queue to the external system.

Figure 9 shows the components involved in this part of the solution.

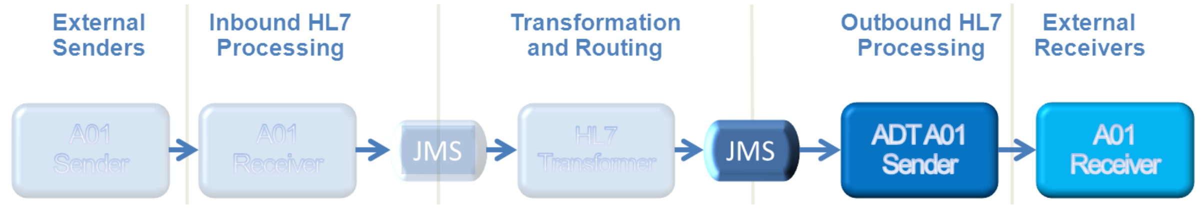


Figure 9, Sending solution components

We will use the CMDHL7Listener command line client to receive HL7 ADT messages and 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.

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

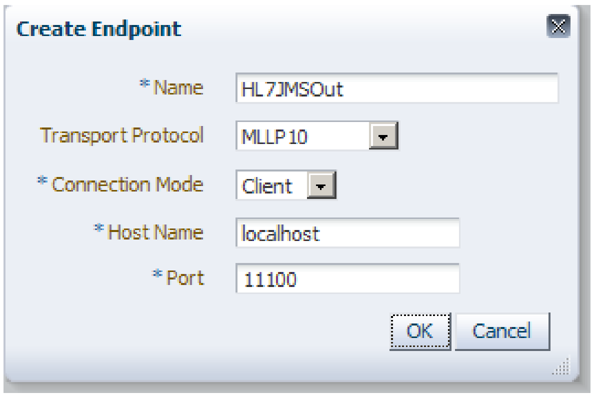
```
java -jar c:\tools\CMDHL7\CMDHL7Listener_v0.7.jar -p 11100 -s
c:\hl7\received
```

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

```
C:\Documents and Settings\Administrator>java -jar
c:\tools\CMDHL7\CMDHL7Listener_v0.7.jar -c ID_ -p 11100 -s
c:\hl7\received
03/02/2013 10:32:24 AM au.id.czapski.hl7.CMDHL7Listener main
INFO: Port: 11100
03/02/2013 10:32:24 AM au.id.czapski.hl7.CMDHL7Listener main
INFO: Store in: c:\hl7\received
03/02/2013 10:32:24 AM ca.uhn.log.HapiLogImpl info
INFO: au.id.czapski.hl7.SimpleACKApplication registered to handle *^*
messages
03/02/2013 10:32:25 AM ca.uhn.log.HapiLogImpl info
INFO: SimpleServer running on port 11100
```

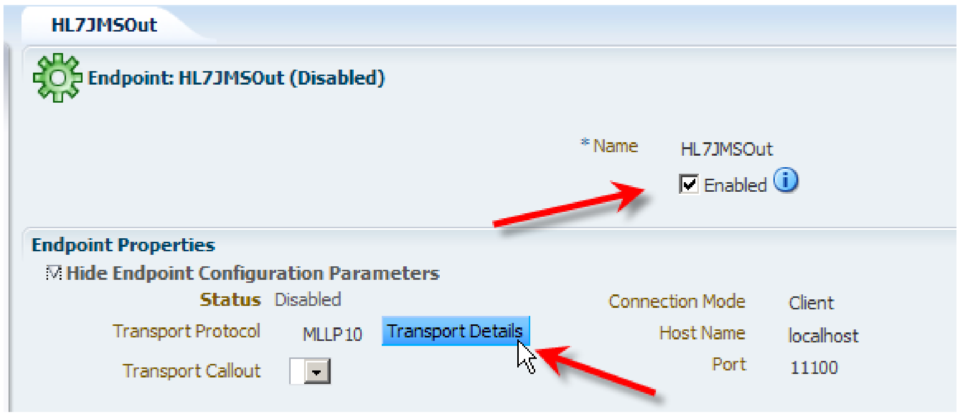
Once the external system is ready to receive messages we can configure the endpoint which will send messages to it.

- ☐ 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 on the "Endpoint" node and choose "Create"
- ☐ Name the endpoint "HL7JMSOut", configure it as a client which uses the MLLP 1.0 protocol and sends to localhost on port 11100, then click "OK"


Create Endpoint

* Name	HL7JMSOut
Transport Protocol	MLLP10
* Connection Mode	Client
* Host Name	localhost
* Port	11100

- ☐ Check the "Enabled" checkbox. When we "Apply" this configuration later, the endpoint will be started.
- ☐ Click the "Transport Details" button


HL7JMSOut

Endpoint: HL7JMSOut (Disabled)

* Name HL7JMSOut

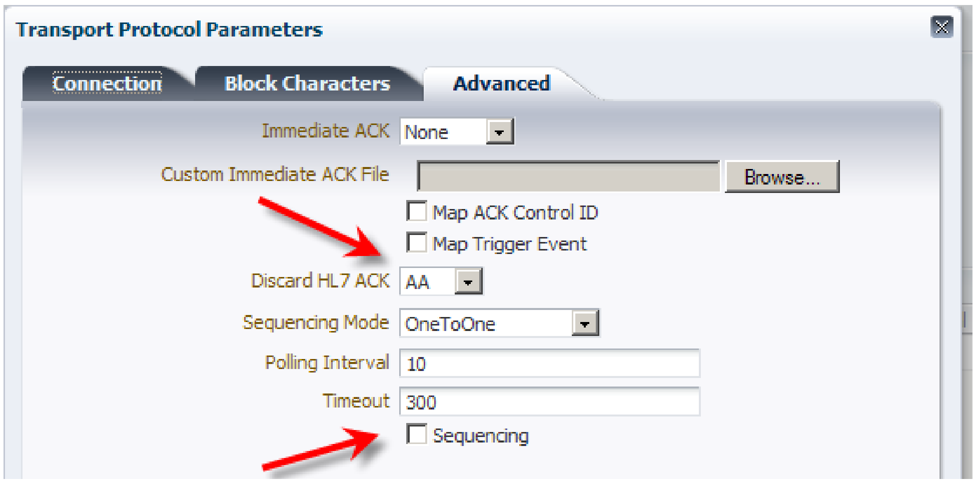
☑ Enabled

Endpoint Properties

Hide Endpoint Configuration Parameters

Status	Disabled	Connection Mode	Client
Transport Protocol	MLLP10	Host Name	localhost
Transport Callout		Port	11100

- ☐ Click the "Advanced" tab in the "Transport Protocol Parameters" dialogue box, set the following properties, and click "OK":
    - Discard ACK: AA
    - Uncheck Sequencing

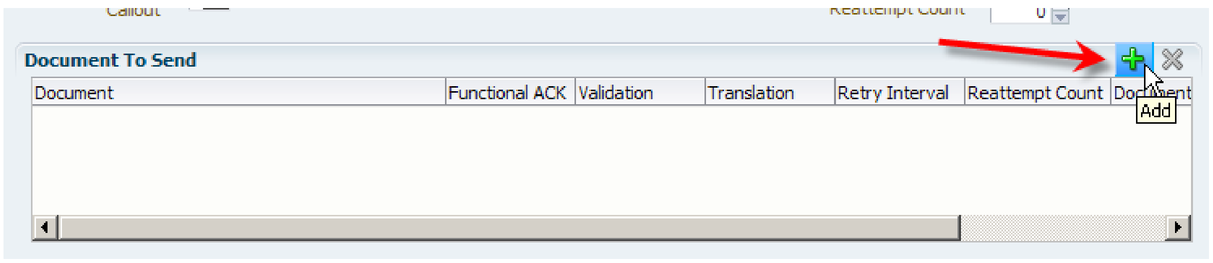


Transport Protocol Parameters

Connection | Block Characters | Advanced

Immediate ACK	None
Custom Immediate ACK File	
	☐ Map ACK Control ID
	☐ Map Trigger Event
Discard HL7 ACK	AA
Sequencing Mode	OneToOne
Polling Interval	10
Timeout	300
	☐ Sequencing

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

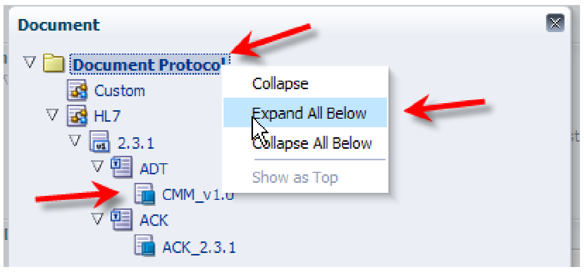


Document To Send

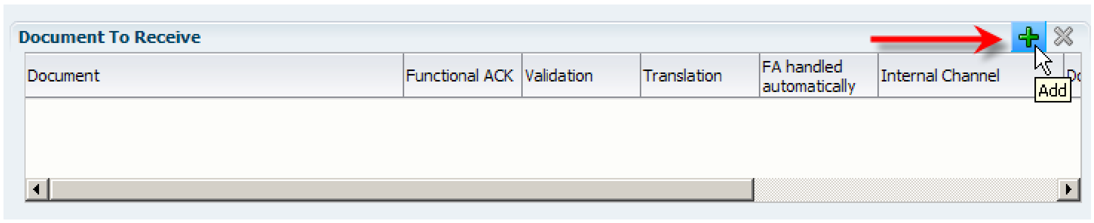
Document	Functional ACK	Validation	Translation	Retry Interval	Reattempt Count	Document

☐ 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

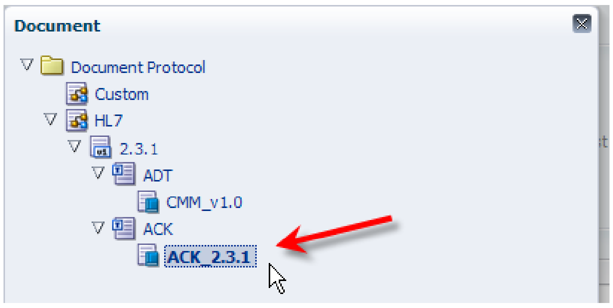


Document To Receive

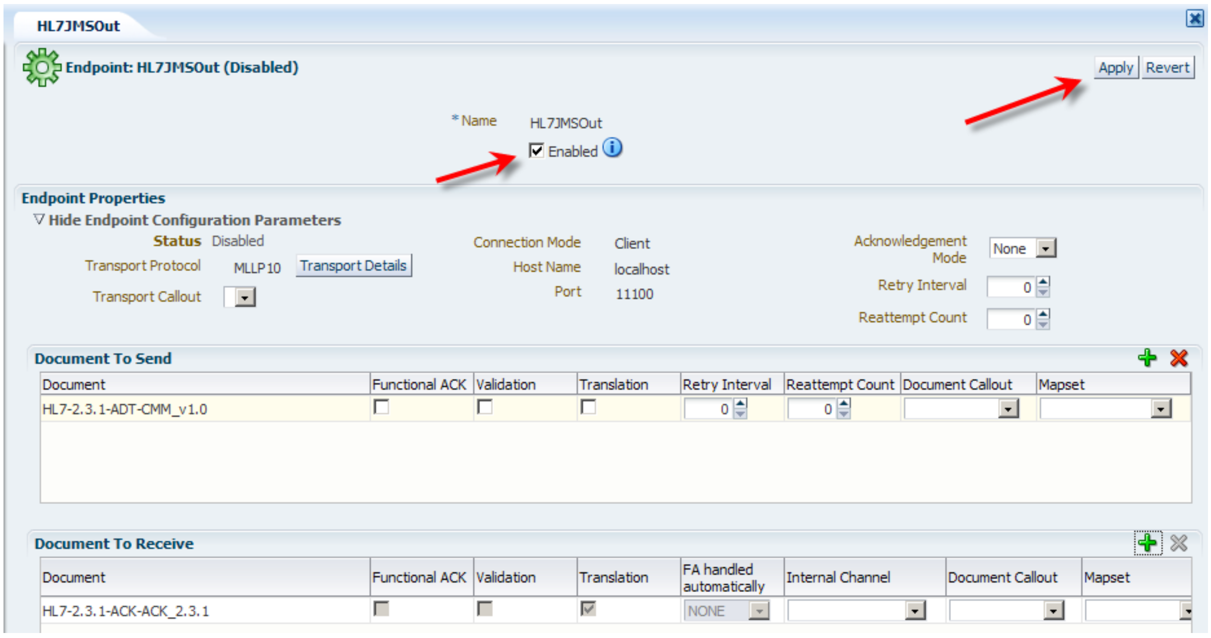
Document	Functional ACK	Validation	Translation	FA handled automatically	Internal Channel	Do

☐ 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



Endpoint: HL7JMSOut (Disabled)

* Name HL7JMSOut
☑ Enabled

Endpoint Properties
Hide Endpoint Configuration Parameters

Status	Disabled	Connection Mode	Client	Acknowledgement Mode	None
Transport Protocol	MLLP10	Host Name	localhost	Retry Interval	0
Transport Callout		Port	11100	Reattempt Count	0

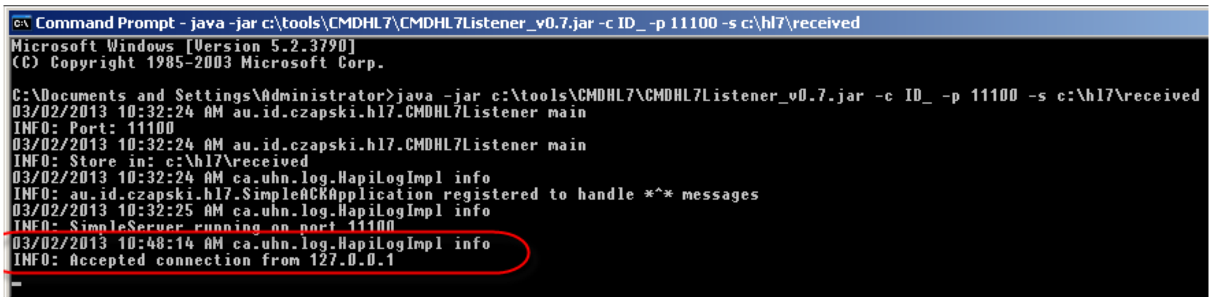
Document To Send

Document	Functional ACK	Validation	Translation	Retry Interval	Reattempt Count	Document Callout	Mapset
HL7-2.3.1-ADT-CMM_v1.0	☐	☐	☐	0	0		

Document To Receive

Document	Functional ACK	Validation	Translation	FA handled automatically	Internal Channel	Document Callout	Mapset
HL7-2.3.1-ACK-ACK_2.3.1	☐	☐	☑	NONE			

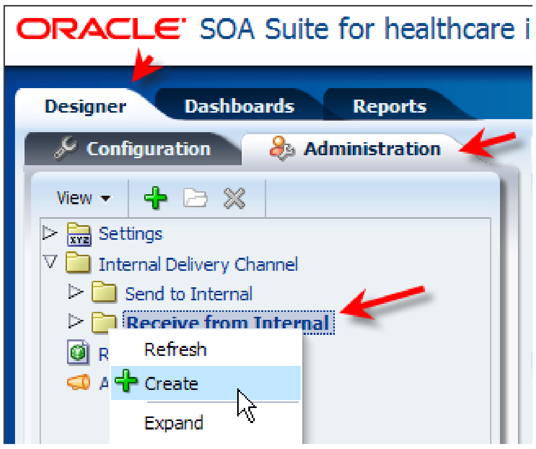
☐ Note the trace in the command window belonging to the listening external


Command Prompt - java -jar c:\tools\CMDHL7\CMDHL7Listener_v0.7.jar -c ID_ -p 11100 -s c:\hl7\received

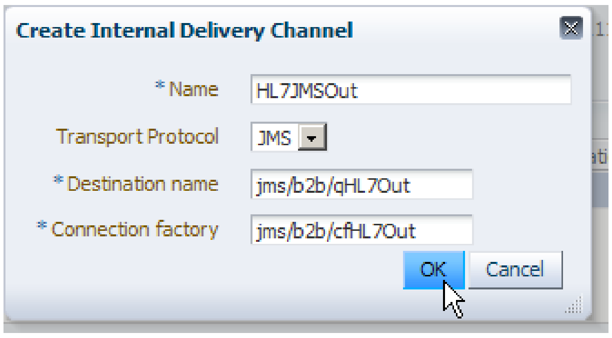
```
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>java -jar c:\tools\CMDHL7\CMDHL7Listener_v0.7.jar -c ID_ -p 11100 -s c:\hl7\received
03/02/2013 10:32:24 AM au.id.czapski.hl7.CMDHL7Listener main
INFO: Port: 11100
03/02/2013 10:32:24 AM au.id.czapski.hl7.CMDHL7Listener main
INFO: Store in: c:\hl7\received
03/02/2013 10:32:24 AM ca.uhn.log.HapiLogImpl info
INFO: au.id.czapski.hl7.SimpleACKApplication registered to handle *^* messages
03/02/2013 10:32:25 AM ca.uhn.log.HapiLogImpl info
INFO: SimpleServer running on port 11100
03/02/2013 10:48:14 AM ca.uhn.log.HapiLogImpl info
INFO: Accepted connection from 127.0.0.1
_
```

As it stands, the sender will only be given messages from the B2B_OUT_QUEUE, if there are any destined for it. What we want to do in this article is to have this endpoint handle messages deposited in the "qHL7Out" JMS queue. To accomplish this we must configure an "Internal Delivery Channel" of the right kind, which will be polled by the "SOA Suite for healthcare integration" infrastructure for messages to process. We will do this in this section.

☐ Click the "Designer" Tab → "Administration" Tab, expand the "Internal Delivery Channel", right click on the "Receive from Internal" node" and choose "Create"

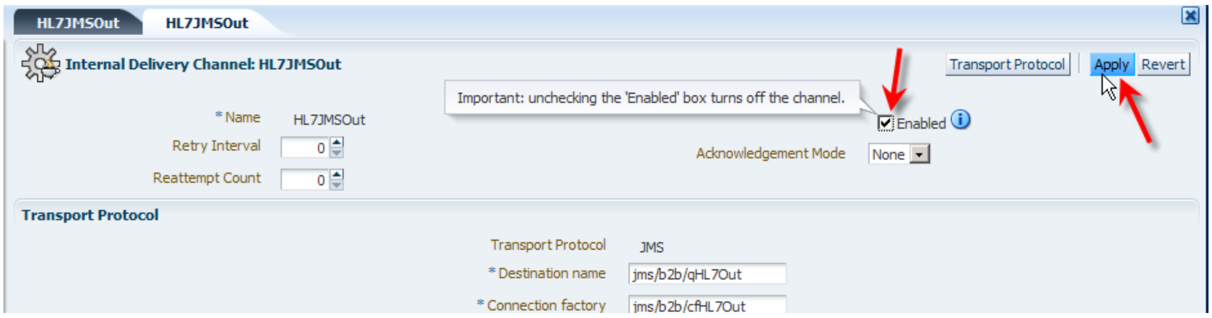


☐ Name the channel "HL7JMSOut", specify "jms/b2b/qHL7Out" as destination name, "jsm/b2b/cfHL7Out" as connection factory and click "OK". Note that these are the JNDI names we defined when we created the JMS destination objects earlier in this article


Create Internal Delivery Channel

* Name	HL7JMSOut
Transport Protocol	JMS
* Destination name	jms/b2b/qHL7Out
* Connection factory	jms/b2b/cfHL7Out

☐ Check the "Enabled" checkbox and click "Apply" to enable this delivery channel


Internal Delivery Channel: HL7JMSOut

Important: unchecking the 'Enabled' box turns off the channel.

* Name	HL7JMSOut
Retry Interval	0
Reattempt Count	0
Enabled	✓
Acknowledgement Mode	None

Transport Protocol

Transport Protocol	JMS
* Destination name	jms/b2b/qHL7Out
* Connection factory	jms/b2b/cfHL7Out

From this point onward any messages which are deposited in the qHL7Out will be delivered to the "SOA Suite for healthcare integration" for processing. Whether it can process these messages successfully will depend on the content of the message and the set of required JMS user-defined properties which the "SOA Suite for healthcare integration" requires to work out which endpoint is to handle the message.

Submit a message to JMS Queue for sending

Recall that we configured the endpoint to not perform translation so the endpoint expects the message it gets to send to already be a HL7 v2 delimited message.

Let's assume that we have a HL7 v2 delimited payload shown below, where the symbol "┘" denotes the carriage return which separates HL7 v2 segments.

```
MSH|^~\&|SystemA|HosA|PI|MDM|2008090801529||ADT^A01|000000_CTLID_2008090801529|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||||||||V2008090801529^^^^VISIT|
||||||||||||||||||||||||2008090801529┘
```

It is very important that carriage returns which separate segments are not converted to new lines (as on Unix) or carriage return+new line (as on Windows). If you are using a text editor use one which can convert line terminators and preserve line terminators, for example Notepad++, which I use.

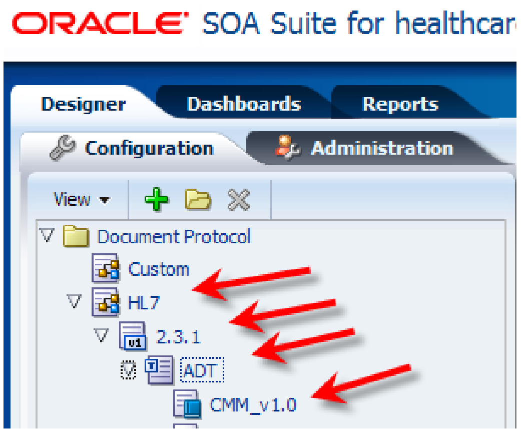
Specific user-defined JMS properties must be set on the message to allow the "SOA Suite for healthcare integration" to figure out which endpoint will handle the message and to figure out what kind of document this message is. The following minimum JMS user-defined properties must be set:

Table 1, JMS User-defined Properties for sending by HL7JMSOut endpoint

Property Name	Property Value
DOCUMENT_PROTOCOL_NAME	HL7
DOCTYPE_REVISION	2.3.1
DOCTYPE_NAME	ADT
DOCUMENT_DEFINITION_NAME	CMM_v1.0
TO_ENDPOINT	HL7JMSOut

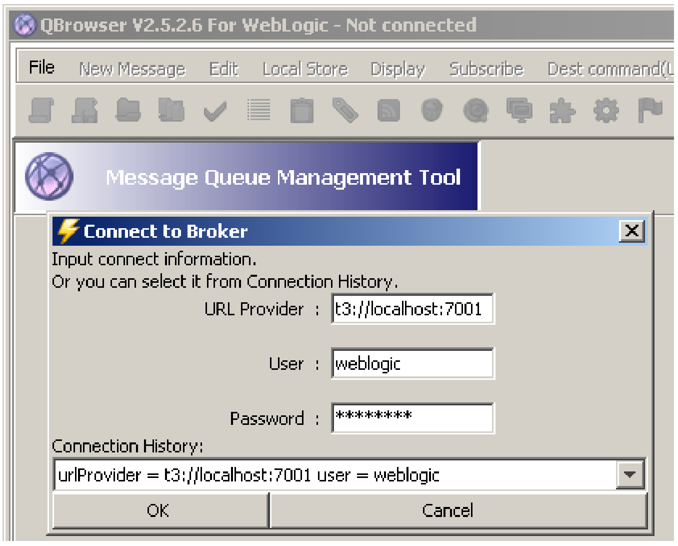
The first four properties allow the "SOA Suite for healthcare integration" to workl out the document type to use. The last property allows it to deliver it to the correct endpoint for sending.

Recall these values form the "Design" → "Configuration" → "Document Protocol" hierarchy.



Let's now use the QBrowser to construct and send this message.

☐ Start the QBrowser application, choose "File" → "New Connection" and log in, perhaps with credentials weblogic/welcome1


QBrowser V2.5.2.6 For WebLogic - Not connected
File New Message Edit Local Store Display Subscribe Dest command(L
Message Queue Management Tool
Connect to Broker

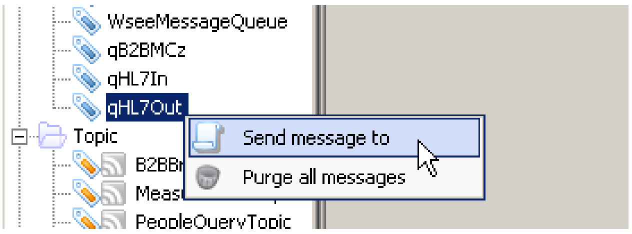
Input connect information.
Or you can select it from Connection History.

URL Provider :	t3://localhost:7001
User :	weblogic
Password :	********

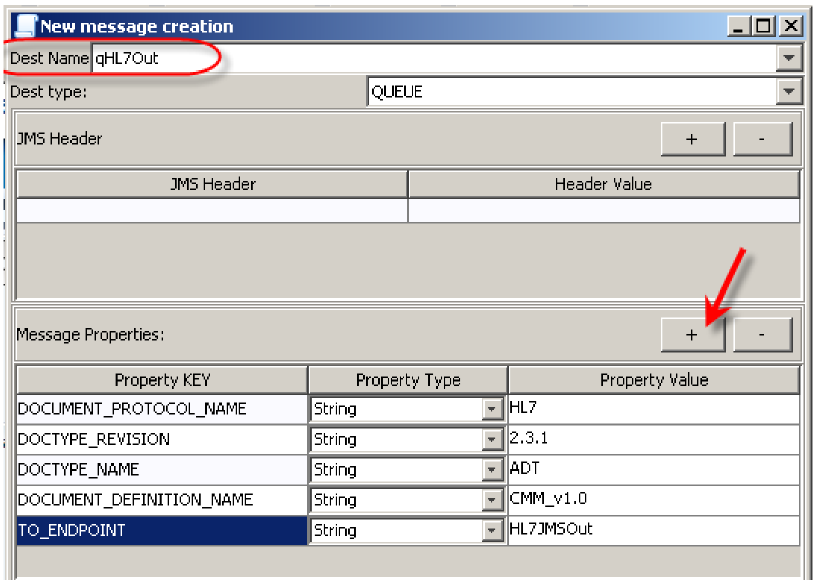
Connection History:
urlProvider = t3://localhost:7001 user = weblogic

☐ Click on the name of the queue, qHL7Out, to select it

☐ Right-click on the name of the queue, qHL7Out, and choose "Send message to"



☐ Use the "Plus" button alongside the "Message Properties" to add five string properties as shown in Table 1

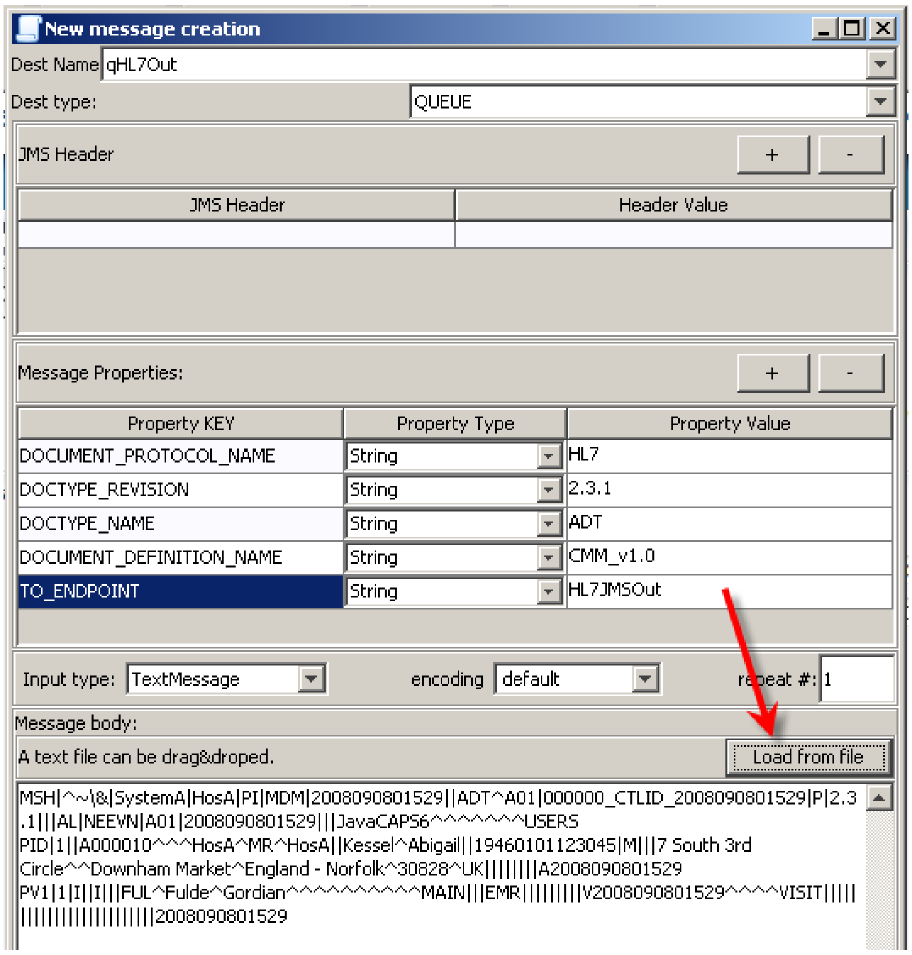


Dest Name qHL7Out
Dest type: QUEUE

JMS Header	Header Value

Property KEY	Property Type	Property Value
DOCUMENT_PROTOCOL_NAME	String	HL7
DOCTYPE_REVISION	String	2.3.1
DOCTYPE_NAME	String	ADT
DOCUMENT_DEFINITION_NAME	String	CMM_v1.0
TO_ENDPOINT	String	HL7JMSOut

☐ If you have the ADT_A01_output1.hl7 file from earlier articles, use the "Load from file" button to locate and load the payload for the message – if not, copy and paste the payload



Dest Name: qHL7Out

Dest type: QUEUE

JMS Header

JMS Header	Header Value

Message Properties:

Property KEY	Property Type	Property Value
DOCUMENT_PROTOCOL_NAME	String	HL7
DOCTYPE_REVISION	String	2.3.1
DOCTYPE_NAME	String	ADT
DOCUMENT_DEFINITION_NAME	String	CMM_v1.0
TO_ENDPOINT	String	HL7JMSOut

Input type: TextMessage encoding: default repeat #: 1

Message body:

A text file can be drag&droped.

```
MSH|^~\&|SystemA|HosA|PI|MDM|2008090801529||ADT^A01|000000_CTLID_2008090801529|P|2.3
.1|||AL|NEEVN|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|||||||||V2008090801529^^^^VISIT|||||
||||||||||||||||||||2008090801529
```

☐ Click "Send", "Send" and "OK"

☐ Click the "Reports" Tab in the healthcare integration console to see message completed in the message tracker

☐ Inspect the message file in the c:\hl7\receive directory, or the directory you configured for the CMDHL7Listener to which to write files

☐ Switch to Healthcare Integration Console, "HL7JMSOut" endpoint configuration, check the "Translation" checkbox and "Apply" – this will cause the endpoint to expect a HL7 v2 XML message, rather than the delimited message we used before, and will cause it to translate the XML message to the delimited message on the way out

☐ Switch to the QBrowser and replace the HL7 v2 delimited payload with the equivalent XML payload, perhaps from the ADT_A01_output1.xml file from earlier articles, use the "Load from file" button to locate and load the payload for the message – if not, copy and paste the payload shown in the code box below

```xml
<?xml version="1.0" encoding="utf-8"?>
<ADT_A01 xmlns="urn:hl7-org:v2xml" Standard="HL7"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:hl7-org:v2xml
C:/GlassFishESBv22/glassfish/domains/domain1/jbi/service-assemblies/HL7A01Delim2XML
/HL7A01Delim2XML-sun-file-binding/sun-file-binding/HL7v231/ADT_A01.xsd">

  <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>
      <TS.1>2008090801529</TS.1>
    </MSH.7>
    <MSH.9>
      <MSG.1>ADT</MSG.1>
      <MSG.2>A01</MSG.2>
    </MSH.9>
    <MSH.10>000000_CTLID_2008090801529</MSH.10>
    <MSH.11>
      <PT.1>P</PT.1>
    </MSH.11>
    <MSH.12>
      <VID.1>2.3.1</VID.1>
    </MSH.12>
    <MSH.15>AL</MSH.15>
    <MSH.16>NE</MSH.16>
  </MSH>
  <EVN>
    <EVN.1>A01</EVN.1>
    <EVN.2>
      <TS.1>2008090801529</TS.1>
    </EVN.2>
    <EVN.5>
      <XCN.1>JavaCAPS6</XCN.1>
      <XCN.8>USERS</XCN.8>
    </EVN.5>
  </EVN>
  <PID>
    <PID.1>1</PID.1>
    <PID.3>
      <CX.1>A000010</CX.1>
      <CX.4>
        <HD.1>HosA</HD.1>
      </CX.4>
      <CX.5>MR</CX.5>
      <CX.6>
        <HD.1>HosA</HD.1>
      </CX.6>
    </PID.3>
    <PID.5>
      <XPN.1>
        <FN.1>Kessel</FN.1>
      </XPN.1>
      <XPN.2>Abigail</XPN.2>
    </PID.5>
    <PID.7>
      <TS.1>19460101123045</TS.1>
    </PID.7>
    <PID.8>M</PID.8>
    <PID.11>
      <XAD.1>7 South 3rd Circle</XAD.1>
      <XAD.3>Downham Market</XAD.3>
      <XAD.4>England - Norfolk</XAD.4>
      <XAD.5>30828</XAD.5>
      <XAD.6>UK</XAD.6>
    </PID.11>
    <PID.19>A2008090801529</PID.19>
  </PID>
  <PV1>
    <PV1.1>1</PV1.1>
    <PV1.2>I</PV1.2>
    <PV1.4>I</PV1.4>
    <PV1.7>
      <XCN.1>FUL</XCN.1>
      <XCN.2>
        <FN.1>Fulde</FN.1>
      </XCN.2>
      <XCN.3>Gordian</XCN.3>
      <XCN.13>MAIN</XCN.13>
    </PV1.7>
    <PV1.10>EMR</PV1.10>
    <PV1.19>
      <CX.1>V2008090801529</CX.1>
      <CX.5>VISIT</CX.5>
    </PV1.19>
    <PV1.44>
      <TS.1>2008090801529</TS.1>
    </PV1.44>
  </PV1>
</ADT_A01>
```

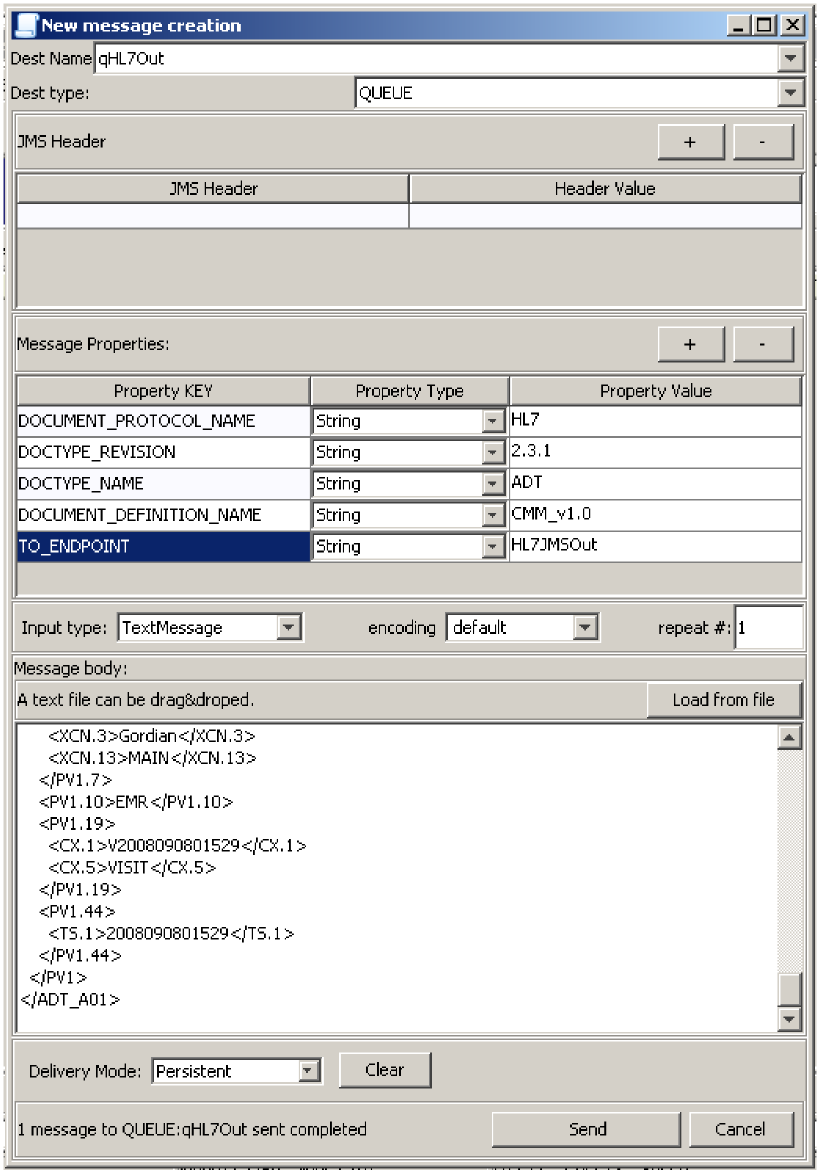
Note the attribute 'Standard="HL7"' at the top of the XML message, as an attribute of the ADT_A01. This is a required attribute which enable the "SOA Suite for healthcare integration" to figure out which protocol it is dealing with. Any XML message which does not carry this attribute will fail (capitalisation is critical as well) with an exception message along the lines of:

```
Error Text Error Brief : XEngine error.
Error Code HC-51507
Error Severity ERROR
Error Level ERROR_LEVEL_COLLABORATION
Error Description Machine Info: (R1PS5HCI) Description: Payload validation error.
```

The application server log will have a somewhat more enlightening message, along the lines of:

```
[2013-02-03T12:55:43.847+11:00] [AdminServer] [TRACE] [] [oracle.soa.hc.engine] [tid: Workmanager:
, Version: 0, Scheduled=false, Started=false, Wait time: 0 ms\r\n] [userId: <anonymous>] [ecid:
c332ab9c10f8b387:-27494294:13c9d91215d:-8000-0000000000001341,0] [SRC_CLASS:
oracle.tip.b2b.system.DiagnosticService] [APP: soa-infra] [SRC_METHOD: synchedLog_J] Notification:
notifyApp: Enqueue the exception message:[[
<Exception xmlns="http://integration.oracle.com/B2B/Exception"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <correlationId>C0A8E9E913C9DC424DD00000202D31FA</correlationId>
  <b2bMessageId>ID:<175441.1359856542910.0></b2bMessageId>
  <errorCode>HC-51507</errorCode>
  <errorText>
  <![CDATA[
Error Brief :
XEngine error.

  ]]>
  </errorText>
  <errorDescription>
  <![CDATA[
Error :
Mandatory attribute missing: Standard.
```

```
    ]]>
    </errorDescription>
    <errorSeverity>2</errorSeverity>
    <errorDetails>
            <parameter name="hc.messageId" value="ID:&lt;175441.1359856542910.0&gt;"/>
            <parameter name="hc.documentTypeName" value="ADT"/>
            <parameter name="hc.documentProtocolVersion" value="2.3.1"/>
            <parameter name="hc.documentDefinitionName" value="CMM_v1.0"/>
            <parameter name="hc.documentProtocolName" value="HL7"/>
            <parameter name="hc.messageType" value="1"/>
            <parameter name="hc.toEndpoint" value="HL7JMSOut"/>
    </errorDetails>
</Exception>
]]
```



Dest Name: qHL7Out
Dest type: QUEUE

JMS Header	Header Value

Property KEY	Property Type	Property Value
DOCUMENT_PROTOCOL_NAME	String	HL7
DOCTYPE_REVISION	String	2.3.1
DOCTYPE_NAME	String	ADT
DOCUMENT_DEFINITION_NAME	String	CMM_v1.0
TO_ENDPOINT	String	HL7JMSOut

Input type: TextMessage encoding: default repeat #: 1

```
   <XCN.3>Gordian</XCN.3>
   <XCN.13>MAIN</XCN.13>
  </PV1.7>
  <PV1.10>EMR</PV1.10>
  <PV1.19>
   <CX.1>V2008090801529</CX.1>
   <CX.5>VISIT</CX.5>
  </PV1.19>
  <PV1.44>
   <TS.1>2008090801529</TS.1>
  </PV1.44>
 </PV1>
</ADT_A01>
```

Delivery Mode: Persistent

1 message to QUEUE:qHL7Out sent completed

☐ Click "Send", "Send" and "OK"

☐ Click the "Reports" Tab in the healthcare integration console to see message completed in the message tracker

☐ Inspect the message file in the c:\hl7\receive directory, or the directory you configured for the CMDHL7Listener to which to write files

☐ If you like, experiment by removing user-defined properties or modifying their values. The user-defined properties we provided are critical to ensuring that the correct document type is used and that the correct endpoint sends the messages.

Configure HL7JMSIn Endpoint

We will now configure the HL7 Inbound endpoint, which will receive HL7 v2 delimited messages and will deliver them, without and with translation, to the JMS Queue named HL7In. Figure 10 highlights the components we will configure.

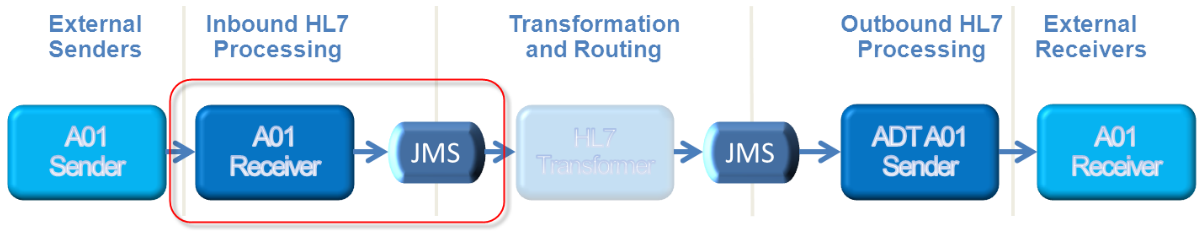


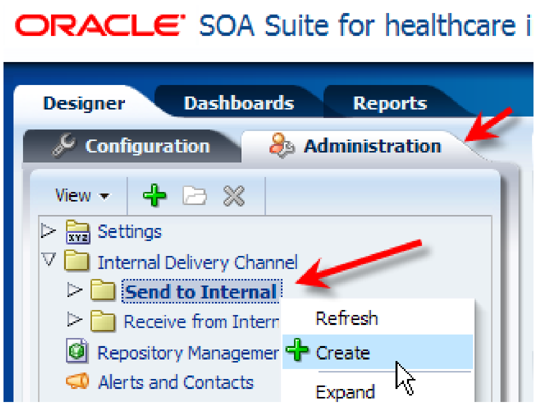
Figure 10, HL7JMSIn solution components

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.

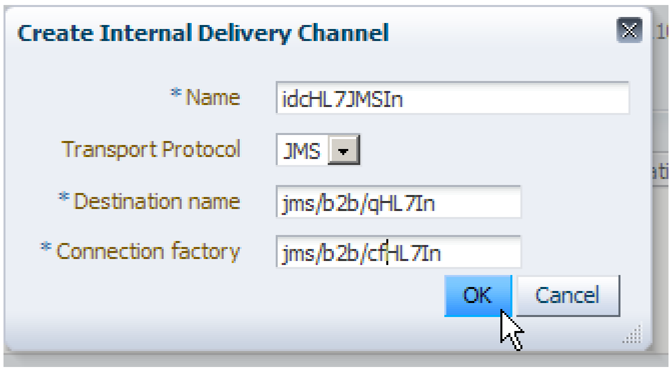
☐ Start Healthcare Integration Console, http://localhost:7001/healthcare, and log in

To make the "SOA Suite for healthcare integration" use a JMS Queue other than the B2B_IN_QUEUE we must create an "Internal Delivery Channel" for the purpose.

☐ Expand "Design" → "Administration" → "Internal Delivery Channels", right-click "Send to Internal" and choose "Create"



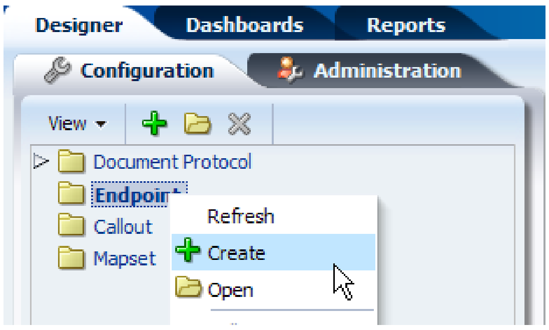
☐ Name the channel "idcHL7JMSIn", provide "jms/b2b/qHL7In" as destination name, provide "jms/b2b/cfHL7In" as connection factory and click "OK" – recall the queue and connection factory JNDI names from earlier in this article


Create Internal Delivery Channel

* Name	idcHL7JMSIn
Transport Protocol	JMS
* Destination name	jms/b2b/qHL7In
* Connection factory	jms/b2b/cfHL7In

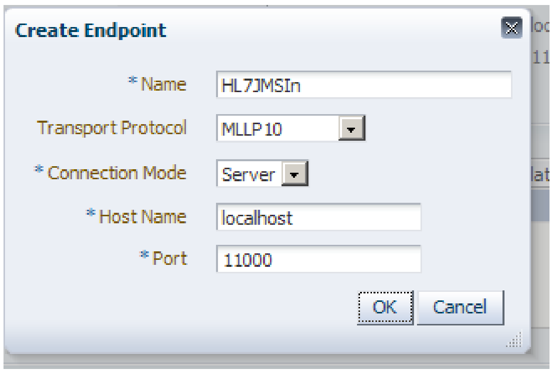
With the internal delivery channel configured we can now configure the endpoint which will use it.

☐ 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: HL7JMSIn
- Transport Protocol: MLLP10
- Connection Mode: server
- Host Name: localhost (or the name of whatever host you are using)
- Port: 11000

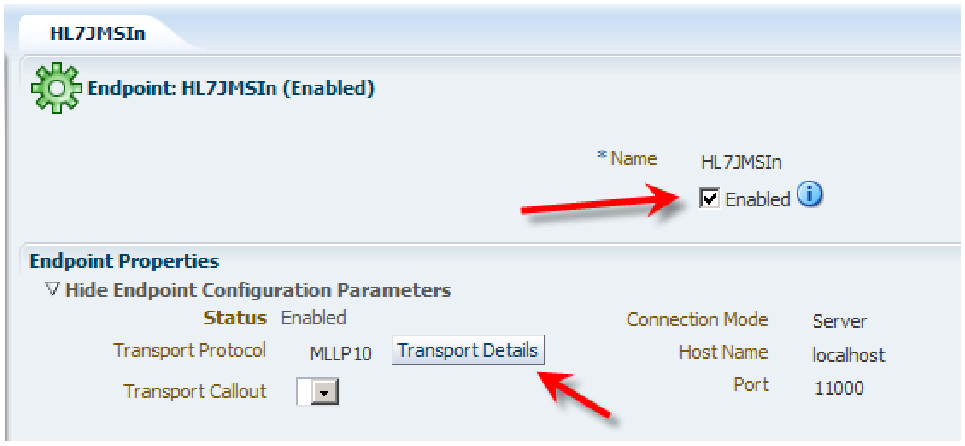

Create Endpoint

* Name	HL7JMSIn
Transport Protocol	MLLP10
* Connection Mode	Server
* Host Name	localhost
* Port	11000

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



HL7JMSIn

Endpoint: HL7JMSIn (Enabled)

* Name HL7JMSIn

☑ Enabled

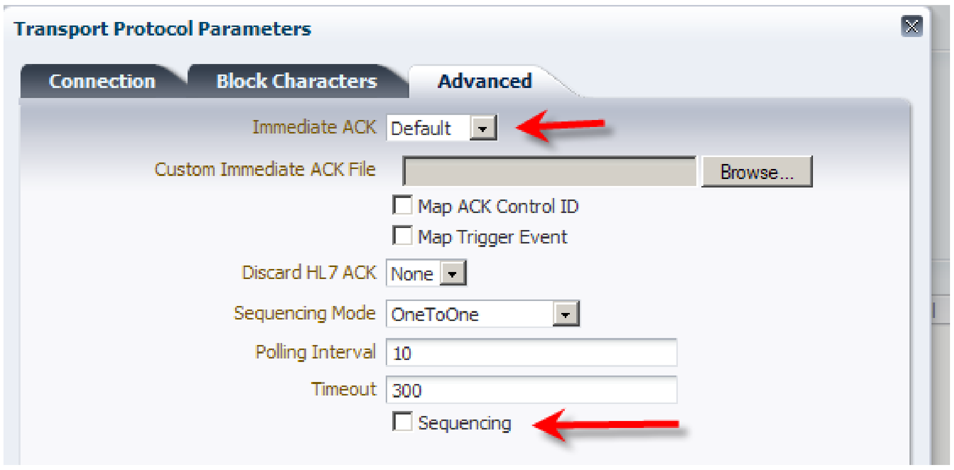
Endpoint Properties

▽ Hide Endpoint Configuration Parameters

Status	Enabled	Connection Mode	Server
Transport Protocol	MLLP10 Transport Details	Host Name	localhost
Transport Callout		Port	11000

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

- Immediate ACK: Default
- Sequencing: Unchecked

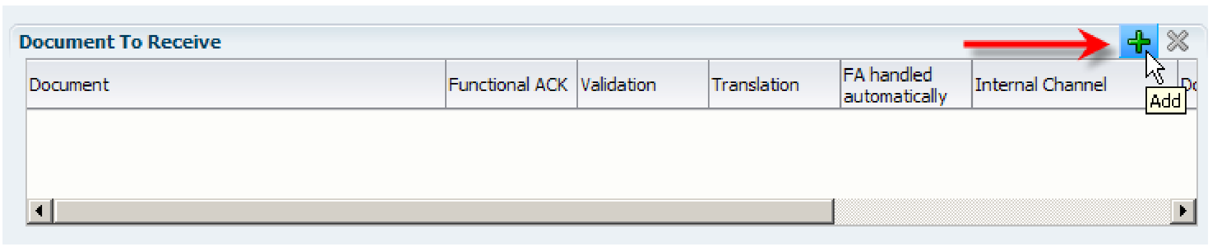


Transport Protocol Parameters

Connection | Block Characters | Advanced

Immediate ACK	Default
Custom Immediate ACK File	Browse...
	☐ Map ACK Control ID
	☐ Map Trigger Event
Discard HL7 ACK	None
Sequencing Mode	OneToOne
Polling Interval	10
Timeout	300
	☐ Sequencing

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

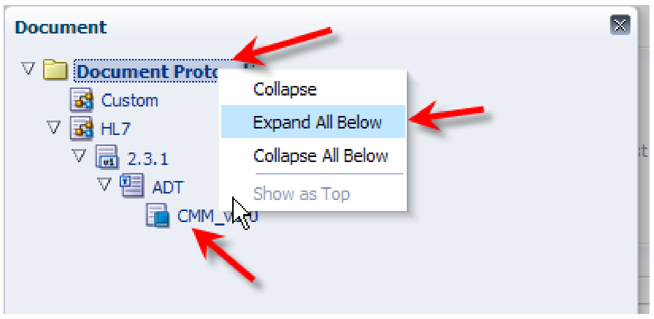


Document To Receive

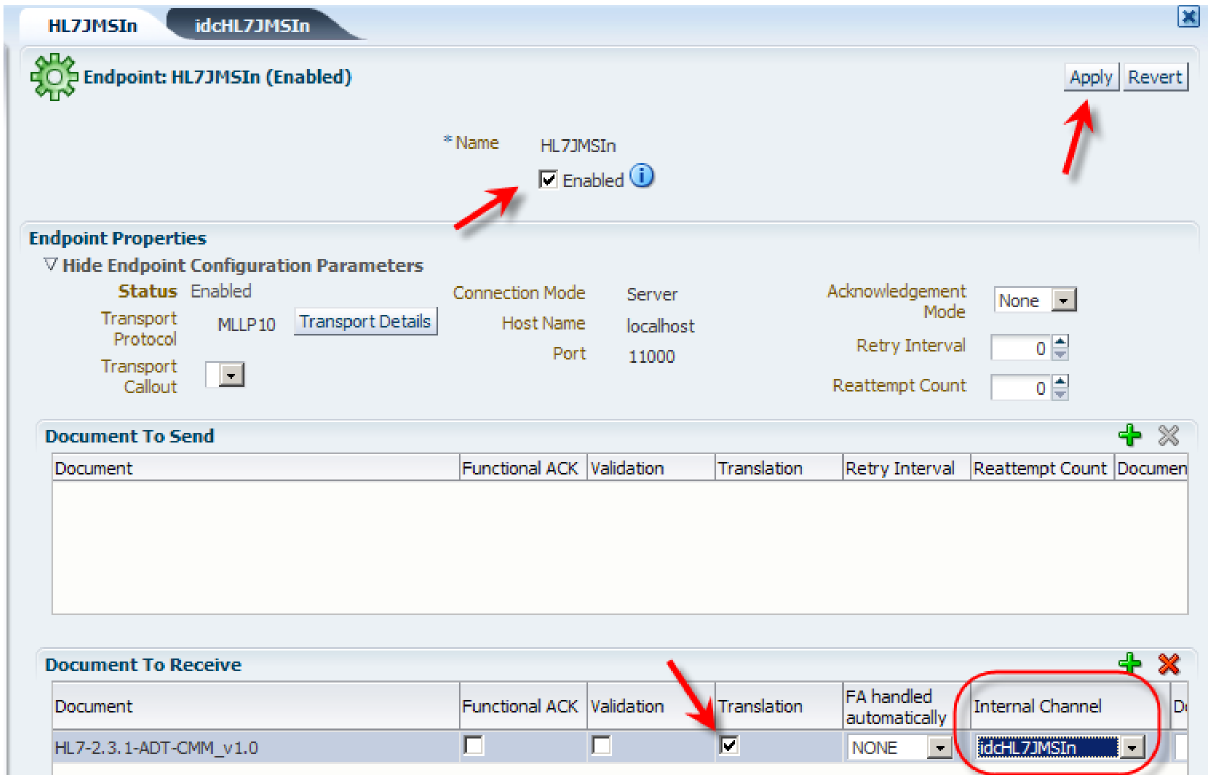
Document	Functional ACK	Validation	Translation	FA handled automatically	Internal Channel

☐ 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"



☐ Uncheck the "Translation" checkbox, choose "idcHL7JMSIn" internal channel, 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



Endpoint: HL7JMSIn (Enabled)

* Name HL7JMSIn

☑ Enabled

Status	Enabled	Connection Mode	Server	Acknowledgement Mode	None
Transport Protocol	MLLP10	Host Name	localhost	Retry Interval	0
Transport Callout		Port	11000	Reattempt Count	0

Document To Send

Document	Functional ACK	Validation	Translation	Retry Interval	Reattempt Count	Documen

Document To Receive

Document	Functional ACK	Validation	Translation	FA handled automatically	Internal Channel	D
HL7-2.3.1-ADT-CMM_v1.0	☐	☐	☑	NONE	idcHL7JMSIn	

☐ Open a command / terminal windows and use the "netstat" command to determine whether the endpoint is running (it behooves us to find out whether the port is used before configuring the port number, and use a different port it 11000 is used)

```
netstat -an | grep 11000
```

or

```
netstat -an | find "11000"
```

The ADT Receiver endpoint is configured and running. It is ready to accept connections and messages. If we now submit a message to this endpoint it will be received and acknowledged, and will be sent to the JMS queue qHL7In.

☐ Use the QBrowser to view messages in the JMS Queue qHL7In – there should be none at this point

We will use the CMDHL7Sender command line client to read a file containing a single HL7 ADT A01 message and submit it to the ADT Receiver endpoint. We will then look at the message and its properties in the JMS Queue qHL7In using the QBrowser tool, and 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 send to the JMS Queue.

☐ 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:

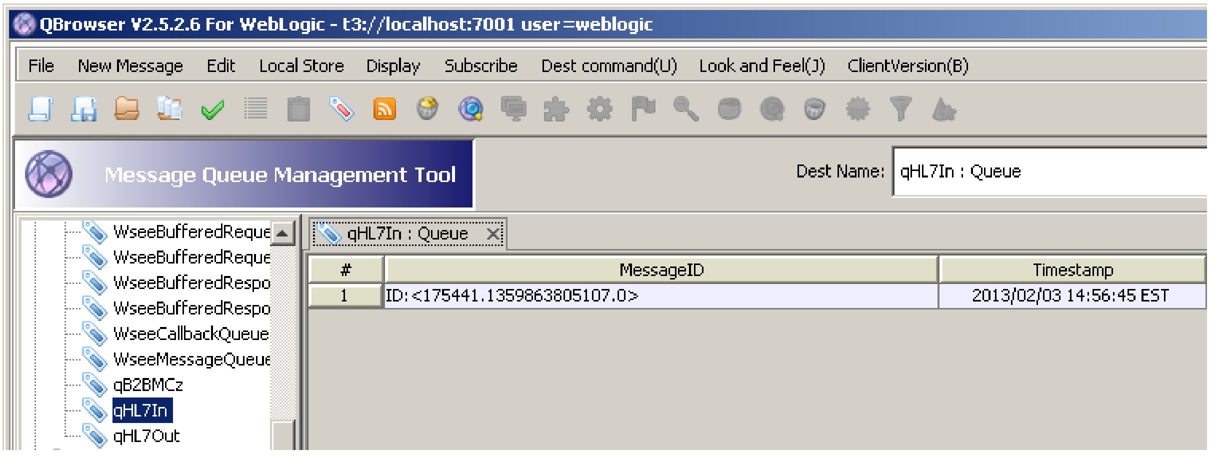
```
MSH|^~\&|SystemA|HosA|PI|MDM|2008090801529||ADT^A01|000000_CTLID_2008
090801529|P|2.3.1|||AL|NE
EVN|A01|2008090801529|||JavaCAPS6^^^^^^^USERS
PID|1||A000010^^^HosA^MR^HosA||Kessel^Abigail||19460101123045|M|||7
South 3rd Circle^^Downham Market^England -
Norfolk^30828^UK|||||||A2008090801529
PV1|1|I||I|||FUL^Fulde^Gordian^^^^^^^^^^MAIN|||EMR||||||||V200809080
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 -n 1
-d \r\r\n -p 11000 -h localhost -t 30000 -f
c:\hl7\adt\sources\ADT_A01_output_1.hl7
```

☐ Refresh the QBrowser display by pressing Ctrl+R (Display → Refresh destination name list) or by selecting a different queue and then selecting the qHL7In again

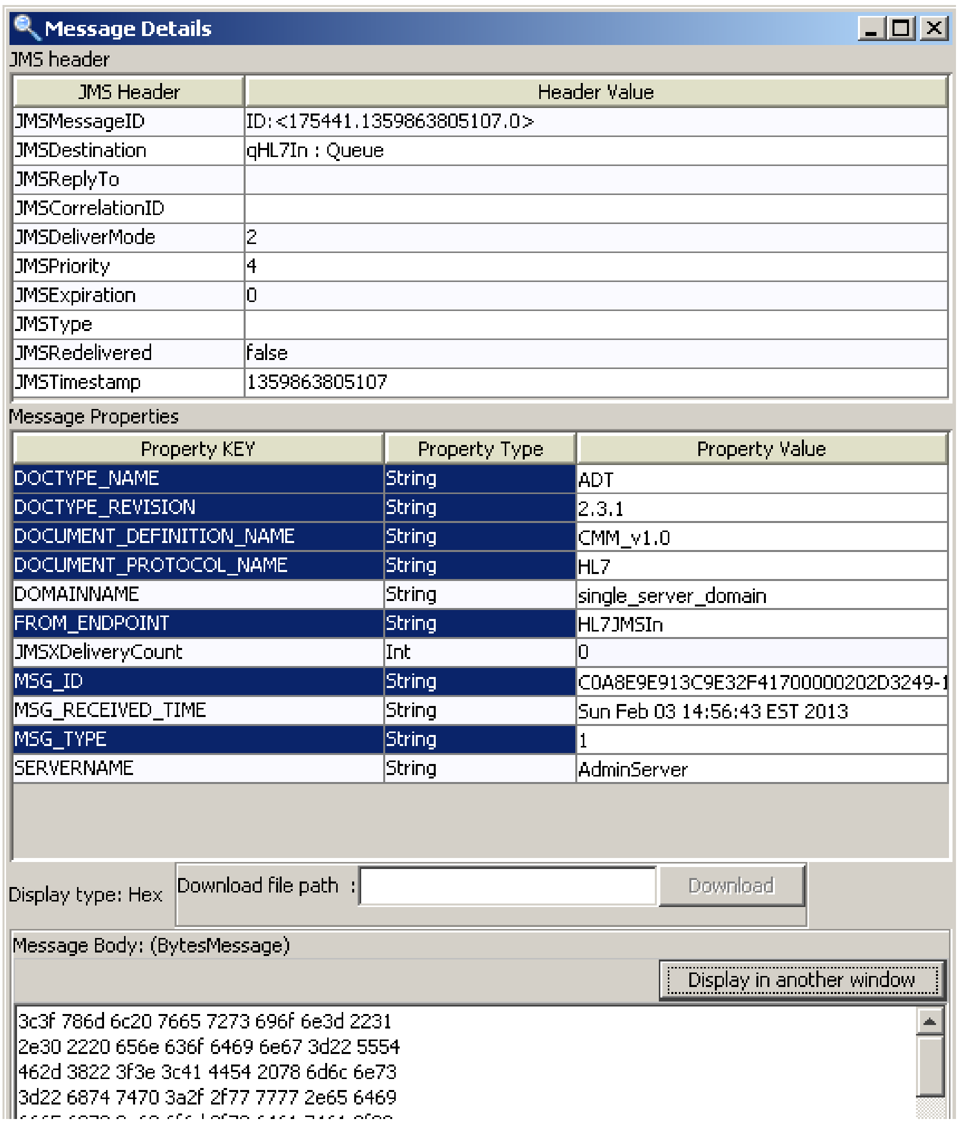
☐ Double-click the message in the window to open it in a dialogue box


QBrowser V2.5.2.6 For WebLogic - t3://localhost:7001 user=weblogic

Dest Name: qHL7In : Queue

#	MessageID	Timestamp
1	ID:<175441.1359863805107.0>	2013/02/03 14:56:45 EST

☐ Review JMS properties, most notably the user-defined properties, and the payload – note that the payload is a binary payload so you can't see the actual content at this point


Message Details

JMS header

JMS Header	Header Value
JMSMessageID	ID:<175441.1359863805107.0>
JMSDestination	qHL7In : Queue
JMSReplyTo	
JMSCorrelationID	
JMSDeliverMode	2
JMSPriority	4
JMSExpiration	0
JMSType	
JMSRedelivered	false
JMSTimestamp	1359863805107

Message Properties

Property KEY	Property Type	Property Value
DOCTYPE_NAME	String	ADT
DOCTYPE_REVISION	String	2.3.1
DOCUMENT_DEFINITION_NAME	String	CMM_v1.0
DOCUMENT_PROTOCOL_NAME	String	HL7
DOMAINNAME	String	single_server_domain
FROM_ENDPOINT	String	HL7JMSIn
JMSXDeliveryCount	Int	0
MSG_ID	String	C0A8E9E913C9E32F41700000202D3249-1
MSG_RECEIVED_TIME	String	Sun Feb 03 14:56:43 EST 2013
MSG_TYPE	String	1
SERVERNAME	String	AdminServer

Display type: Hex

Download file path :

Message Body: (BytesMessage)

```
3c3f 786d 6c20 7665 7273 696f 6e3d 2231
2e30 2220 656e 636f 6469 6e67 3d22 5554
462d 3822 3f3e 3c41 4454 2078 6d6c 6e73
3d22 6874 7470 3a2f 2f77 7777 2e65 6469
```

Note that the critical property "TO_ENDPOINT", which was required for the outbound sender to determine the endpoint, is not present. All other critical properties, dealing with the document name and its hierarchy are there. We could pass this message to the outbound queue as soon as we added the TO_ENDPOINT property with the correct value, alas, QBrowser does not allow us to add properties to the existing message so we need to develop and piece of logic to read from the inbound queue, qHL7In, add the TO_ENDPOINT property and write the message with the modified properties to qHL7Out. We will do this next.

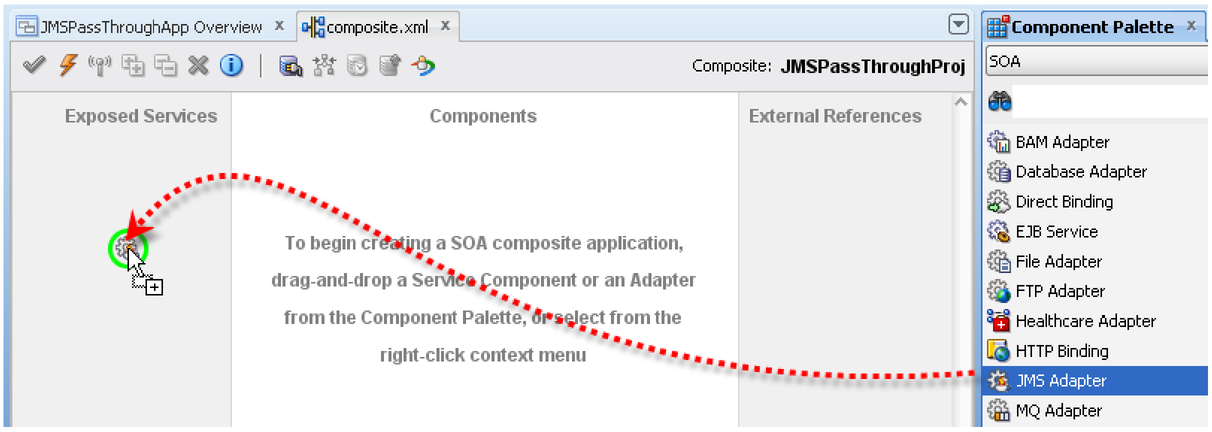
Implement Pass-through Composite Application

With the endpoint configured and deployed we will now develop the SOA Composite to read the message from the JMS queue qHL7In, add a user-defined property TO_ENDPOINT, and write it to the JMS queue qHL7Out.

☐ Start JDeveloper Studio

☐ Create a new SOA application, "JMSPassThroughApp" and a new project "JMSPassThroughProj" with an empty composite – we did this often enough in this article series to not require detailed steps, right?

☐ Drag the "JMS Adapter" from the "Component Palette" → "SOA" → "Service Adapters" list to the "Exposed Services" swim line


JMSPassThroughApp Overview
composite.xml
Composite: JMSPassThroughProj
Component Palette
SOA
Exposed Services
Components
External References
To begin creating a SOA composite application, drag-and-drop a Service Component or an Adapter from the Component Palette, or select from the right-click context menu
BAM Adapter
Database Adapter
Direct Binding
EJB Service
File Adapter
FTP Adapter
Healthcare Adapter
HTTP Binding
JMS Adapter
MQ Adapter

☐ Name the service "HL7In" and click "Next"

☐ Choose "WebLogic JMS" as "oracle Enterprise Messaging Service" and click "Next"

☐ Choose the appserver connection and click "Next"

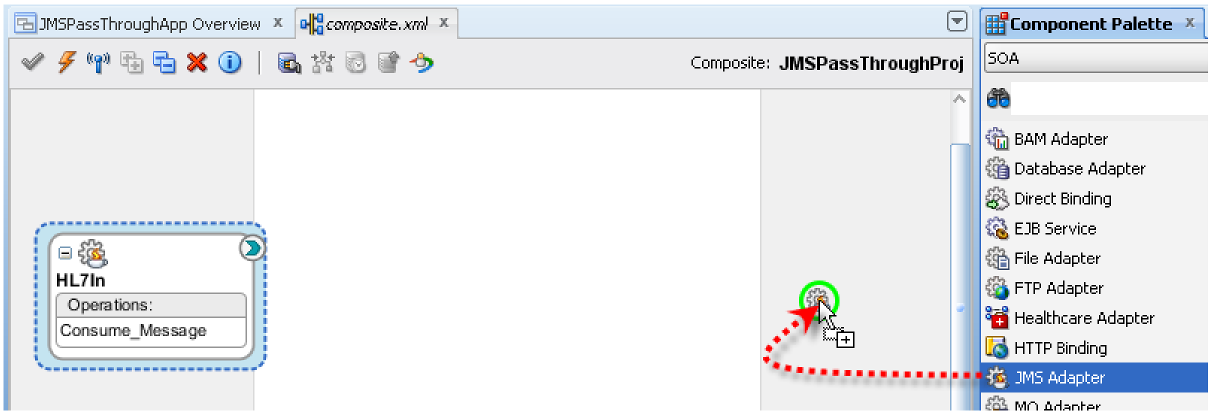
☐ Accept the default for "Adapter Schema" and click "Next"

☐ Choose "Consume Message" "Operation Type" and click "Next"

☐ Choose "jms/b2b/qHL7In" destination name, "jms/b2b/cfHL7In" connection factory, "BytesMessage" for message body type and click "Next"

☐ Check the "Native format translation is not required" check box, then click "Next" and "Finish"

☐ Drag the "JMS Adapter" from the "Component Palette" → "SOA" → "Service Adapters" list to the "External References" swim line



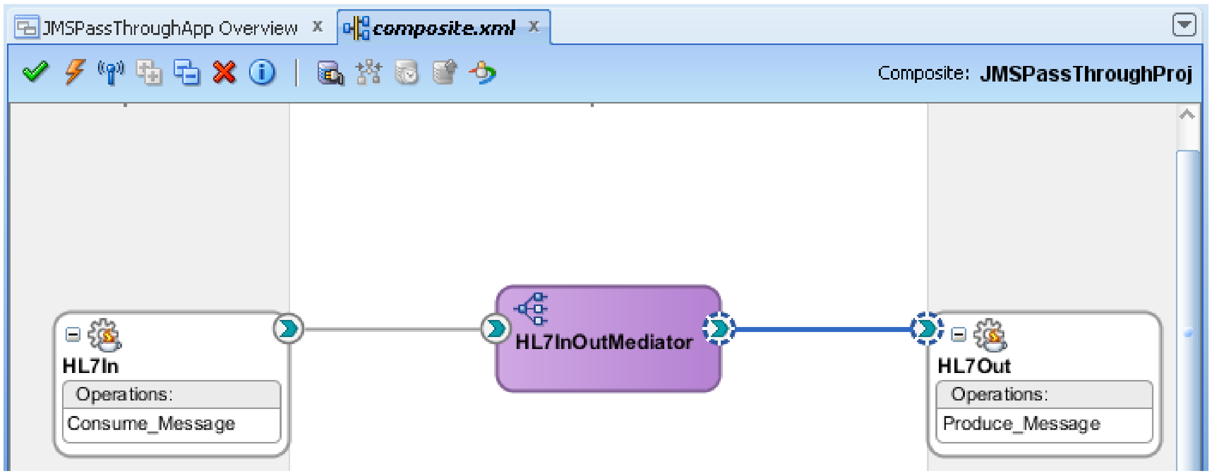
☐ Name the service "HL7Out" and click "Next"

☐ Choose "WebLogic JMS" as "oracle Enterprise Messaging Service" and click "Next"

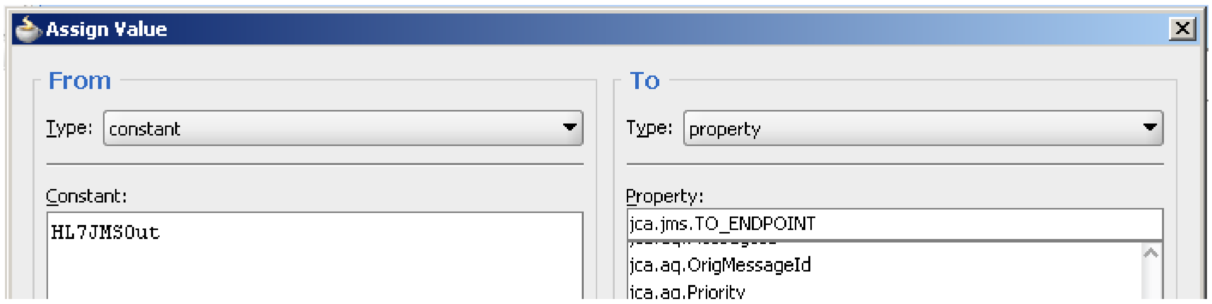
☐ Choose the appserver connection and click "Next"

☐ Accept the default for "Adapter Schema" and click "Next"

- ☐ Choose "Produce Message" as "Operation Type" and click "Next"
- ☐ Choose "jms/b2b/qHL7Out" destination name, "jms/b2b/cfHL7Out" connection factory, "BytesMessage" for message body type and click "Next"
- ☐ Check the "Native format translation is not required" check box, then click "Next" and "Finish"
- ☐ Drag the "Mediator" component from the "Component Palette" → "SOA" → "Service Components" to the "Components" swim line
- ☐ Name the mediator "HL7InOutMediator"
- ☐ Connect the adapters to the mediator component



- ☐ Double-click the mediator component to open the mplan and add the following value assignments
  - ○ Constant "HL7JMSOut" to property "jca.jms.JMSProperty.TO_ENDPOINT"


Assign Value

From

Type: constant

Constant:

HL7JMSOut

To

Type: property

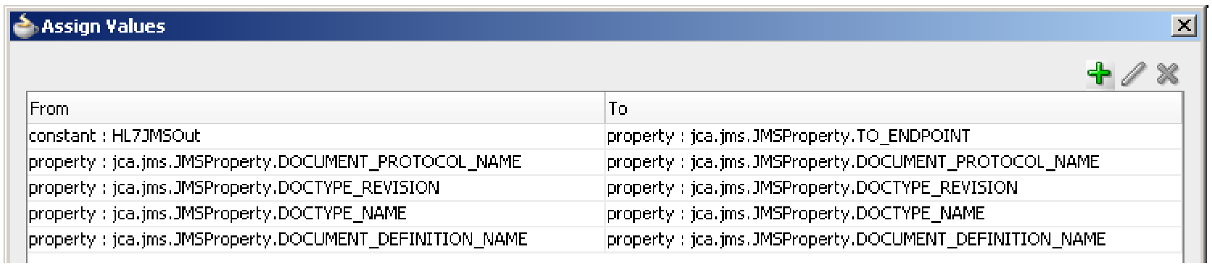
Property:

jca.jms.TO_ENDPOINT

jca.aq.OrigMessageId

jca.aq.Priority

  - ○ From Property "jca.jms.JMSProperty.DOCUMENT_PROTOCOL_NAME" to To Property "jca.jms.JMSProperty.DOCUMENT_PROTOCOL_NAME"
  - ○ From Property "jca.jms.JMSProperty.DOCTYPE_REVISION" to To Property "jca.jms.JMSProperty.DOCTYPE_REVISION"
  - ○ From Property "jca.jms.JMSProperty.DOCTYPE_NAME" to To Property "jca.jms.JMSProperty.DOCTYPE_NAME"
  - ○ From Property "jca.jms.JMSProperty.DOCUMENT_DEFINITION_NAME" to To Property "jca.jms.JMSProperty.DOCUMENT_DEFINITION_NAME"



From	To
constant : HL7JMSOut	property : jca.jms.JMSProperty.TO_ENDPOINT
property : jca.jms.JMSProperty.DOCUMENT_PROTOCOL_NAME	property : jca.jms.JMSProperty.DOCUMENT_PROTOCOL_NAME
property : jca.jms.JMSProperty.DOCTYPE_REVISION	property : jca.jms.JMSProperty.DOCTYPE_REVISION
property : jca.jms.JMSProperty.DOCTYPE_NAME	property : jca.jms.JMSProperty.DOCTYPE_NAME
property : jca.jms.JMSProperty.DOCUMENT_DEFINITION_NAME	property : jca.jms.JMSProperty.DOCUMENT_DEFINITION_NAME

- ☐ Click "Save All", close the "mplan" editor
- ☐ Deploy the project
- ☐ Submit more messages to see the solution work

Summary

In this article we developed and exercise an inbound-to-JMS and JMS-to-outbound HL7 v2 delimited message processing solutions to demonstrate how "SOA Suite for healthcare integration" HL7 messaging endpoints can be used in a SEDA environment where payloads are passed between components via JMS destinations. This is the architecture employed by Java CAPS for this purpose. This model can also be used in ESB-based solutions where JMS destinations are used as persistence points.