Oracle SOA Suite 11g R1 PS5

SOA Suite for healthcare integration Series

HL7 v2 Inbound to File Solution

michael@czapski.id.au

November 2012

Table of Contents

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.

This article works through the mechanics of configuring the “SOA Suite for healthcare integration” to receive a HL7 v2.3.1 ADT message as a Canonical Message and configuring the SOA Suite to write each message to a file in the file system.

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/wp-content/uploads/2012/08/SOASuiteHCI_ch2_Dev_Environ_v0.1.0.pdf.

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. Eventually, towards the end of this article, ADT A01 and ADT A03 messages, used in this solution, 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/wp-content/uploads/2012/09/SOASuiteHCI_ch5_CanonicalMessage_v0.1.0.pdf. The inbound SOA Suite for healthcare integration adapter will perform this casting activity while translating the message from HL7 v2 delimited to the "equivalent" XML format.

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

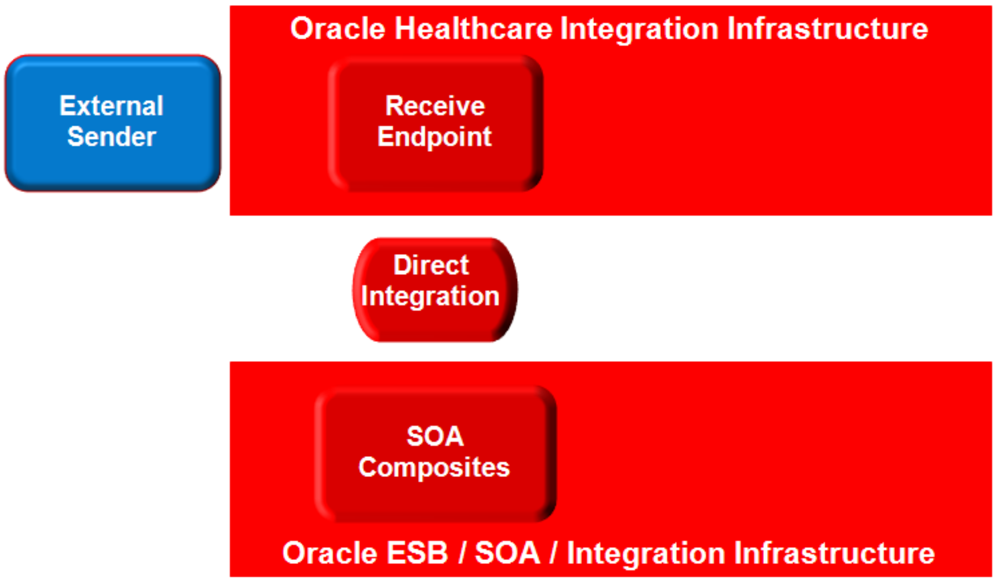

Oracle Healthcare Integration Infrastructure
External Sender
Receive Endpoint
Direct Integration
SOA Composites
Oracle ESB / SOA / Integration Infrastructure

Figure 1 Runtime Components of the Solution

The components and the relationships are discussed in the article "", to be found at ???. To summarise:

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

2. 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
3. Direct Integration is the behind-the-scenes mechanism which hands messages over to an appropriate SOA Suite-based logic component for further processing
4. ESB / SOA / Integration infrastructure hosts the SOA Composites and other logic components which process messages, whether HL7 v2 or not. In our solution it will write each message to a separate file.

The solution components are depicted in Figure 2.

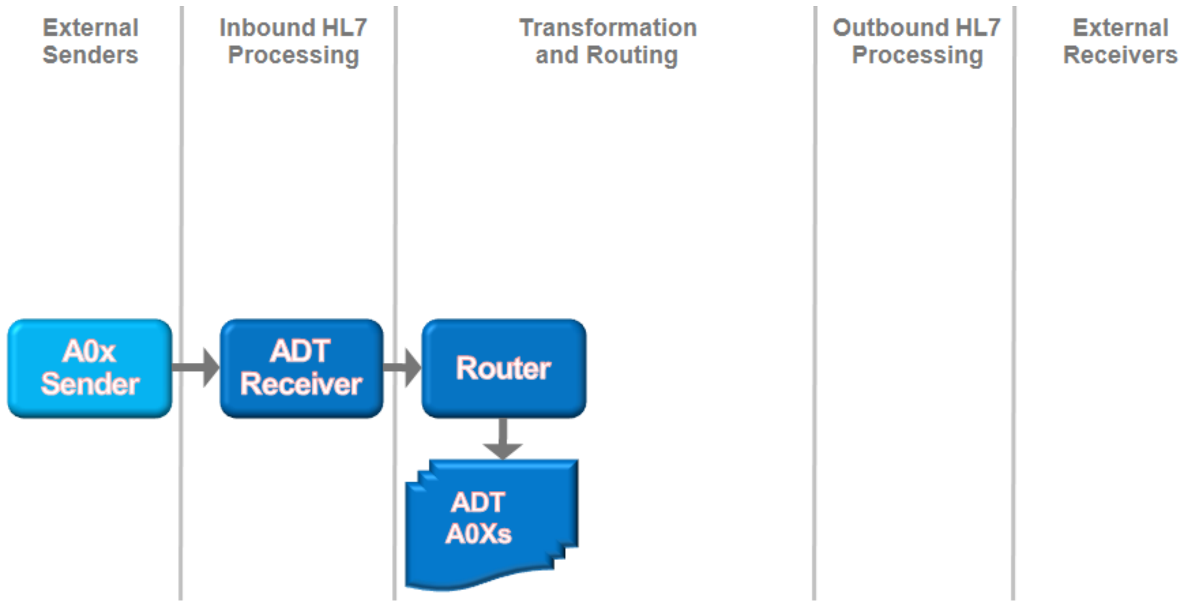


Figure 2 Solution Components

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 writes it to a file.

The solution will be developed in stages, adding (some) complexity and exploring relevant features as we go along.

First, a solution will simply receive a HL7 v2 message, acknowledge it with an immediate acknowledgement and write it to a file with a generated name 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.

In the next stage the solution will be re-configured to transform the HL7 v2 Delimited message to its "equivalent" HL7 v2 XML format and acknowledge it using a Functional Acknowledgement. This acknowledgement will be sent only after the message is parsed by the inbound adapter and validated. If the message passes validation a positive acknowledgement will be sent. If message validation fails a negative acknowledgment will be sent.

Finally, the solution will be modified to set the name of the file to a string which uses message content and messaging environment attribute values, for example HL7 message type and the document type. This variant does not have anything to do with HL7 messaging but illustrates how message content and messaging environment attribute values can be accessed if needed in real-world solutions.

HL7 v2 Inbound to file – delimited message pass-through

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.

It is assumed that the CMM_v1.0.ecs and CMM_v1.0.xsd message structure files, developed in the article "SOA Suite for healthcare integration Series - Creating a Canonical HL7 v2 Message Model", to be found at http://blogs.czapski.id.au/wp-content/uploads/2012/09/SOASuiteHCI_ch5_CanonicalMessage_v0.1.0.pdf are available, since they will be required in this step. If they are not available please go back tp that article and follow the steps to create them.

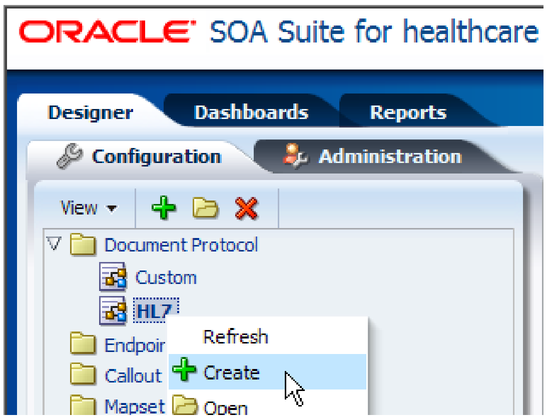
- ☐ Start the Healthcare Integration Console application in your favourite web browser – http://localhost:7001/healthcare.
- ☐ Log in with administrative credentials, for example weblogic/welcome1.

Add CMM_v1.0 Document to Document Protocol Hierarchy

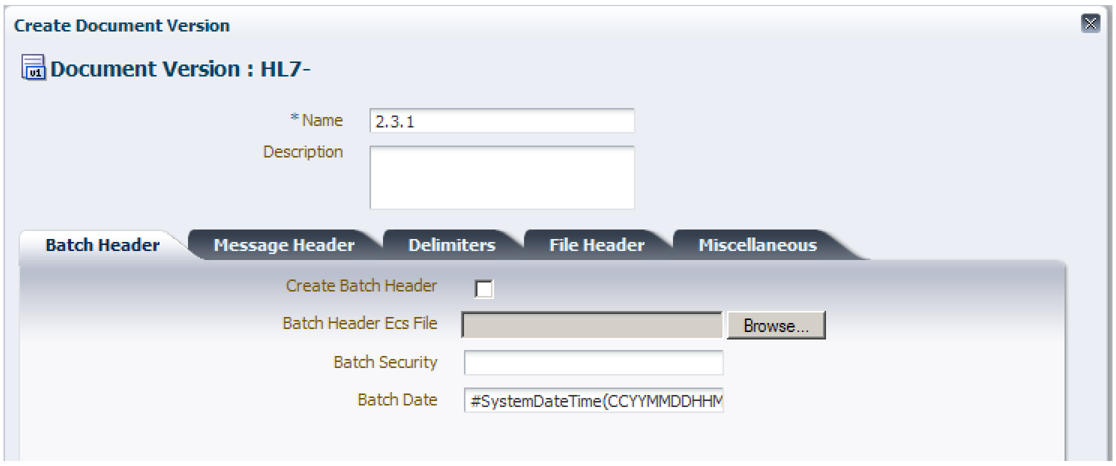
It may sound strange to users of other H7 messaging environments but SOA Suite for healthcare integration uses the term "Document" to describe a HL7 message structure and the message that such a structure describes. The meaning is normally clear in context – we defined documents (message structures) and receive/send documents (messages).

To be able to deal with HL7 messages of a particular kind (HL7 documents of a particular kind) the SOA Suite for healthcare integration must be configured to recognise such messages (documents) and to parse them if required using the correct structure (document) definition. This is what we will do in this section.

- ☐ Expand "Document Protocol" in the "Configuration" tab, right-click "HL7" and choose "Create"



☐ Enter "2.3.1" as the value of the Name field in the "Create Document Version" dialogue box and click "OK".

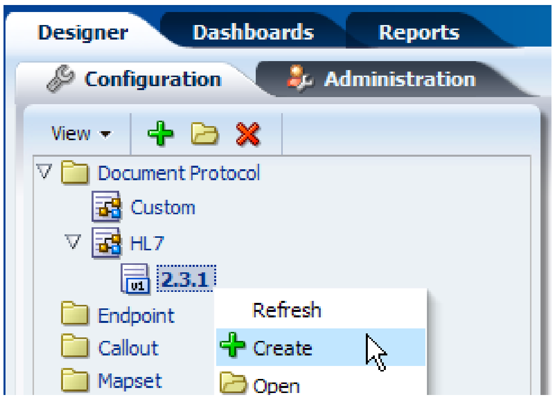

Create Document Version

Document Version : HL7-

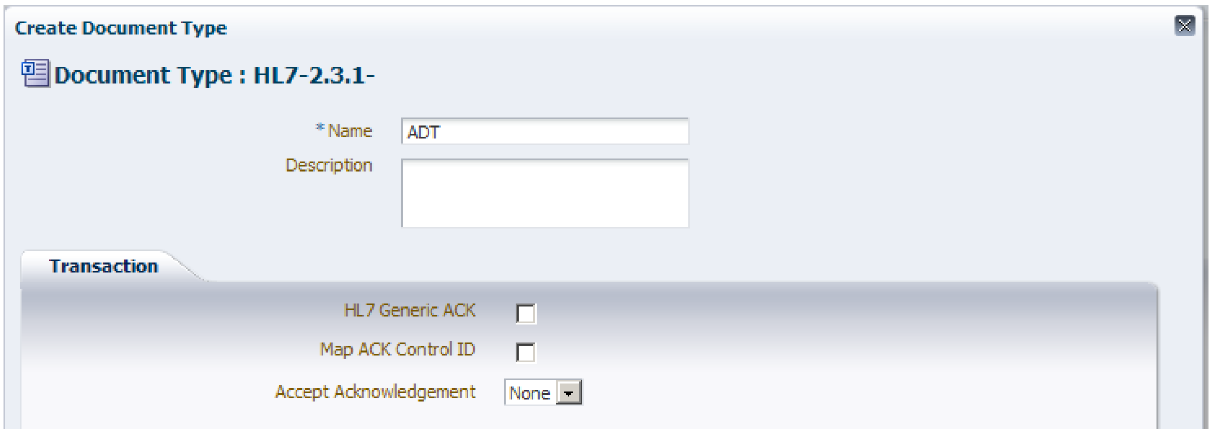
* Name	2.3.1
Description	

Create Batch Header	☐
Batch Header Ecs File	
Batch Security	
Batch Date	#SystemDateTime(CCYYMMDDHHM

☐ Expand the "HL7" node, right-click the new "2.3.1" node and choose "Create"



☐ Enter "ADT" as the value of the Name filed in the "Create Document Type" dialog box and click "OK"


Create Document Type

Document Type : HL7-2.3.1-

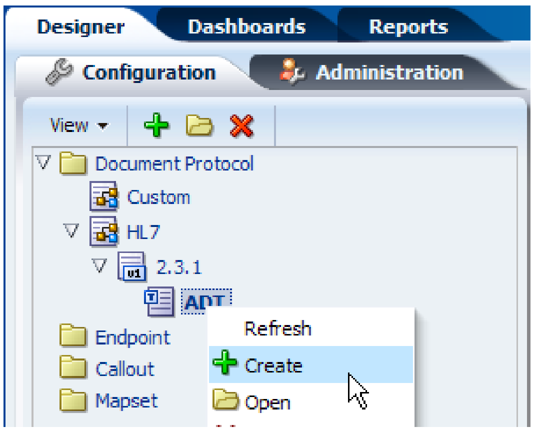
* Name	ADT
Description	

Transaction

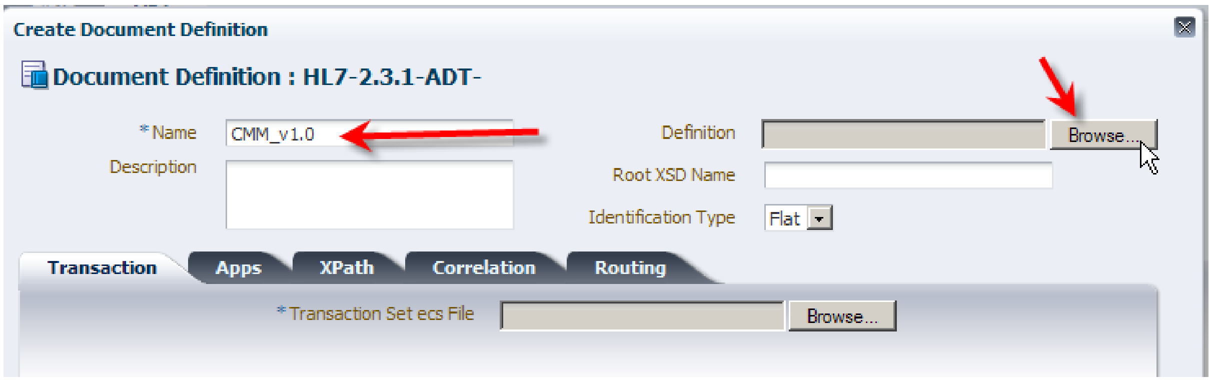
HL7 Generic ACK	☐
Map ACK Control ID	☐
Accept Acknowledgement	None

☐ Expand the node “2.3.1”, right click the new node “ADT” and choose “Create”

A short explanation is in order. The “ADT” message type is a “Generic Message Type”. Normally one would define specific message types, for example ADT_A01, ADT_A03, and so on, and the SOA Suite for healthcare integration would expect and match such messages with appropriate types. Conversely, message of type ADT_A01 message will not match message type ADT_A03 consequently SOA Suite for healthcare integration will not find the document to use to parse it. To allow us to parse multiple ADT messages using a single document type we exploit SOA Suite for healthcare integration’s administrative runtime configuration option – “Generic Message Type” – which when checked allows generic message type configuration to be used for messages for which specific message type has not been defined. We will verify the setting of this configuration option later in this article.



☐ Enter “CMM_v1.0” into the Name field of the “Create Document ” dialog box, click the “Browse” button alongside the “Definition” label, locate the XML Schema Definition file, CMM_v1.0.xsd, and select it

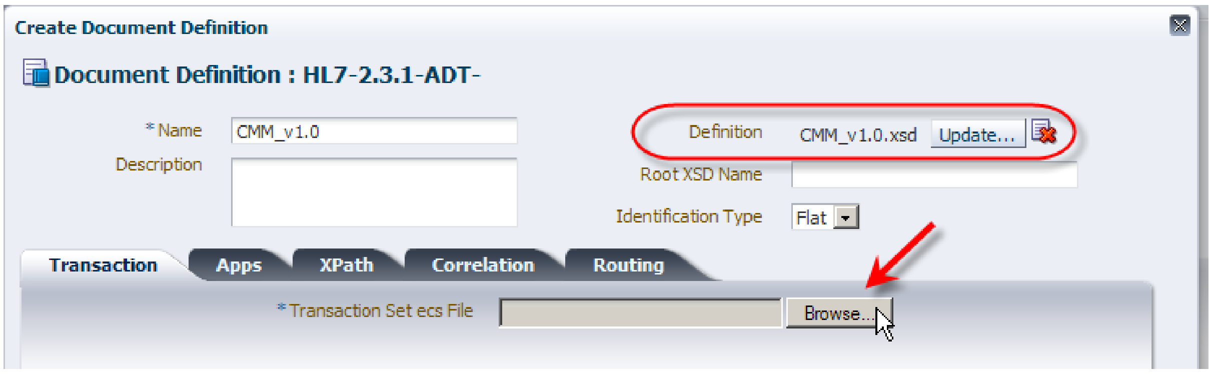

Create Document Definition

Document Definition : HL7-2.3.1-ADT-

* Name	CMM_v1.0	Definition	Browse...
Description		Root XSD Name	
		Identification Type	Flat

* Transaction Set ecs File | Browse...

☐ Click the “Browse” button alongside the “Transaction Set ecs File” label, locate the ECS file, CMM_v1.0.ecs, and select it


Create Document Definition

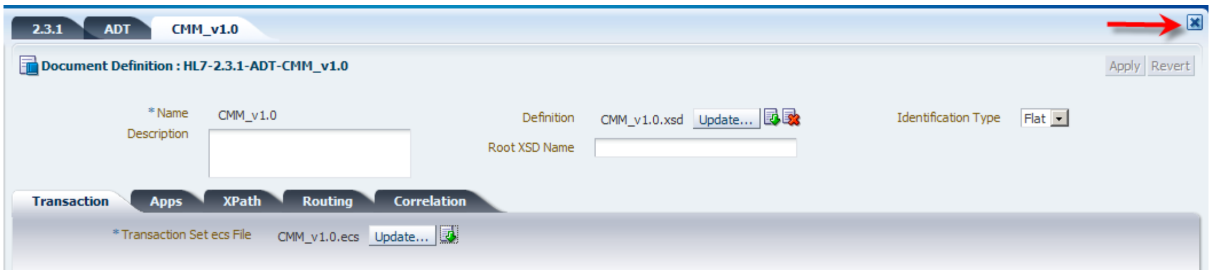
Document Definition : HL7-2.3.1-ADT-

* Name	CMM_v1.0	Definition	CMM_v1.0.xsd Update...
Description		Root XSD Name	
		Identification Type	Flat

* Transaction Set ecs File | Browse...

☐ Click “OK” to complete the dialogue

☐ Use the “Close” “Button” to close the “ADT”, “2.3.1” and “CMM_v1.0” Tabs – you will find the Healthcare Integration Console more responsive with fewer open tabs

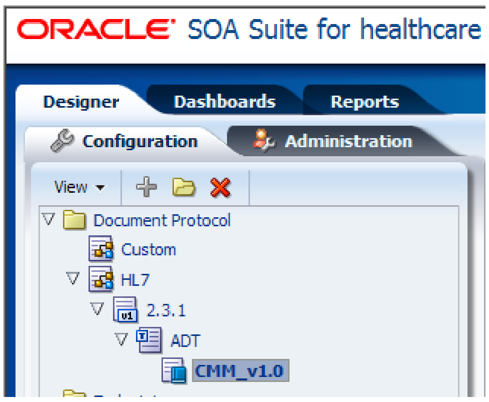


Document Definition : HL7-2.3.1-ADT-CMM_v1.0

* Name	CMM_v1.0	Definition	CMM_v1.0.xsd Update...	Identification Type	Flat
Description		Root XSD Name			

* Transaction Set ecs File | CMM_v1.0.ecs Update...

Our document hierarchy should now look like that shown in the illustration.



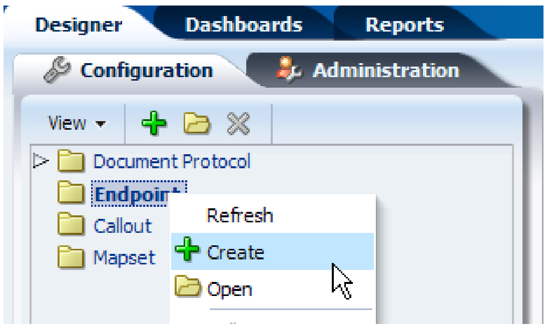
We will use other document types in subsequent articles. We could have "introduced" the all at this time and saved ourselves the time later. For simplicity we will work through step-by-step, configuring components as we need them.

Configure ADT Receiver

We the HL7 v2.3.1 ADT canonical message defined we are in a position to configure the adapter / endpoint which will receive messages of this type.

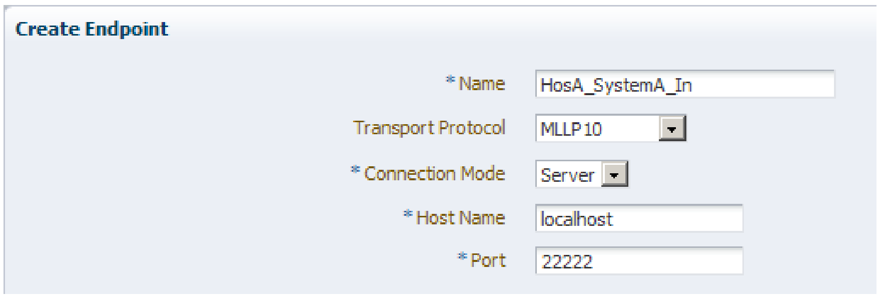
Typically, and in our solution necessarily, the receiver will be a TCP/IP listener supporting the MLLP 1.0 encapsulation protocol. Such a receiver typically "listens" for connections on a specific TCP/IP port, accepts a connection, establishes a session with the sender, receives messages sent to it, sends acknowledgements back over the same connection, and stops when the partner closes the session / connection. SOA Suite for healthcare integration uses the term "Endpoint" to describe both a receiver/listener and a sender. We will use the same term to reduce confusion. In the listening / receiving mode the endpoint is configured as a "server".

☐ 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: HosA_SystemA_In
- Transport Protocol: MLLP10
- Connection Mode: server
- Host Name: localhost (or the name of whatever host you are using)
- Port: 22222



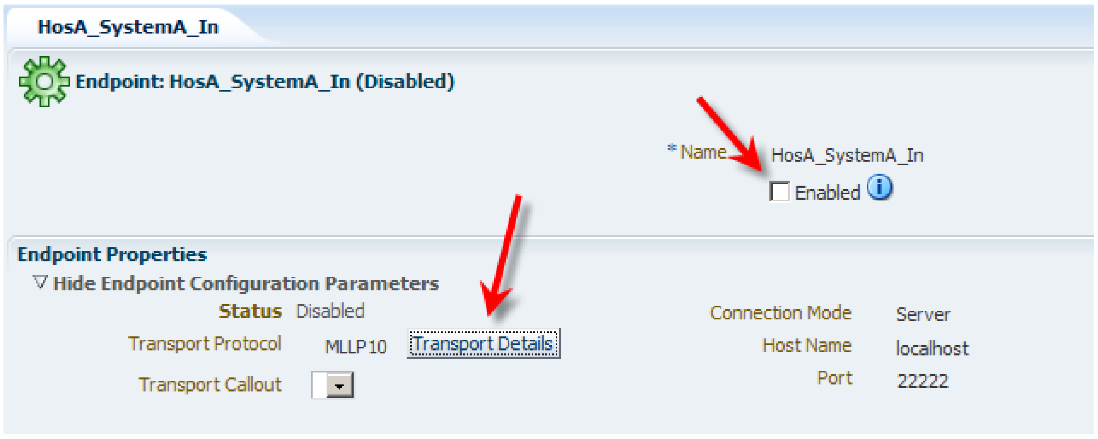
Create Endpoint

* Name	HosA_SystemA_In
Transport Protocol	MLLP10
* Connection Mode	Server
* Host Name	localhost
* Port	22222

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



HosA_SystemA_In

Endpoint: HosA_SystemA_In (Disabled)

* Name HosA_SystemA_In

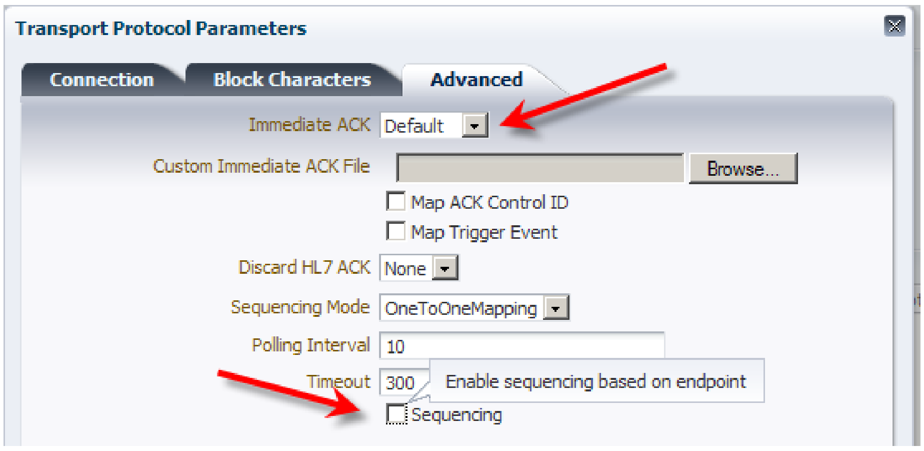
☐ Enabled

Endpoint Properties

▽ Hide Endpoint Configuration Parameters

Status	Disabled	Connection Mode	Server
Transport Protocol	MLLP10 Transport Details	Host Name	localhost
Transport Callout		Port	22222

- ☐ Click the "Advanced" tab in the "Transport Protocol Parameters" dialogue box, set the following properties, and click "OK":
  - Immediate ACK: Default
  - Sequencing: Unchecked

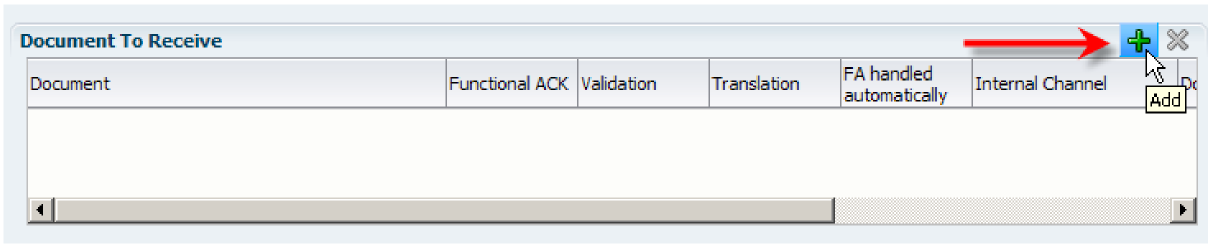


Transport Protocol Parameters

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

Enable sequencing based on endpoint

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

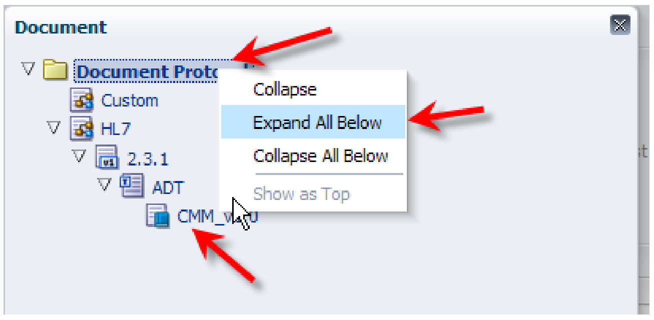


Document To Receive

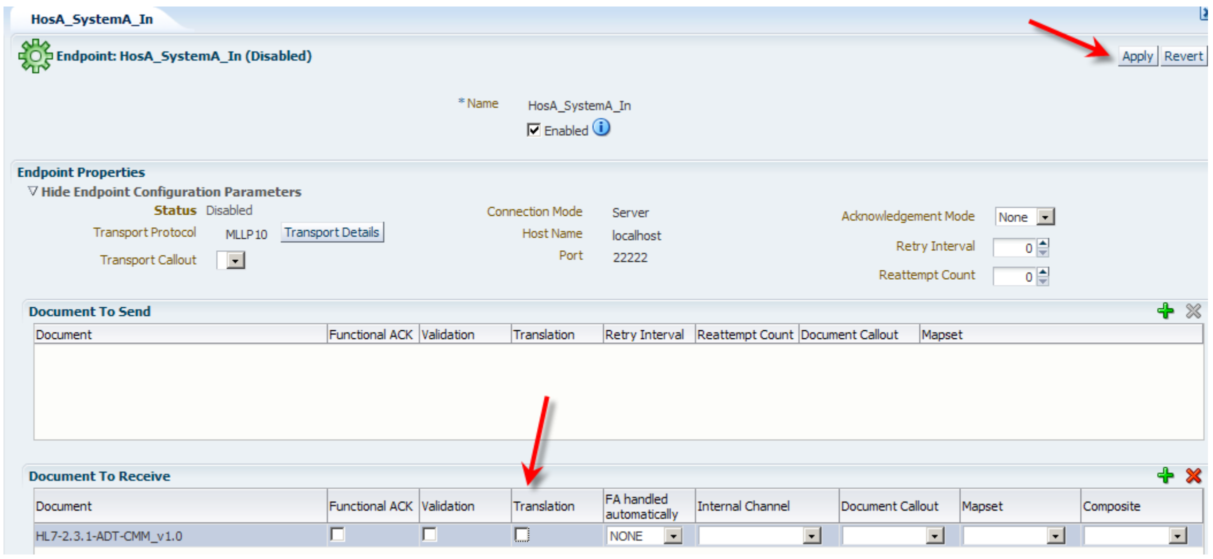
Document	Functional ACK	Validation	Translation	FA handled automatically	Internal Channel	Do

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

- ☐ Select the "CMM_v1.0" document in the HL7→2.3.1→ADT 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


HosA_SystemA_In

Endpoint: HosA_SystemA_In (Disabled)

Apply Revert

* Name	HosA_SystemA_In
	☑ Enabled

Endpoint Properties

▽ Hide Endpoint Configuration Parameters

Status	Disabled	Connection Mode	Server	Acknowledgement Mode	None
Transport Protocol	MLLP10 Transport Details	Host Name	localhost	Retry Interval	0
Transport Callout		Port	22222	Reattempt Count	0

Document To Send

Document	Functional ACK	Validation	Translation	Retry Interval	Reattempt Count	Document Callout	Mapset

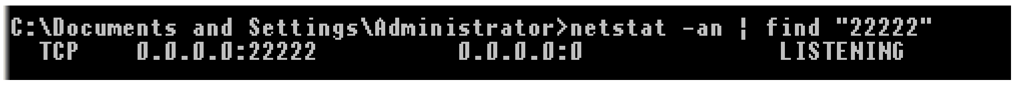
Document To Receive

Document	Functional ACK	Validation	Translation	FA handled automatically	Internal Channel	Document Callout	Mapset	Composite
HL7-2.3.1-ADT-CMM_v1.0	☐	☐	☐	NONE				

☐ 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 22222 is used)

```
netstat -an | grep 22222
```
or

```
netstat -an | find "22222"
```


```
C:\Documents and Settings\Administrator>netstat -an | find "22222"
  TCP    0.0.0.0:22222          0.0.0.0:0              LISTENING
```

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, but will not go anywhere useful because we don't have the other part of the solution, the file writer SOA Composite.

Create Router Composite

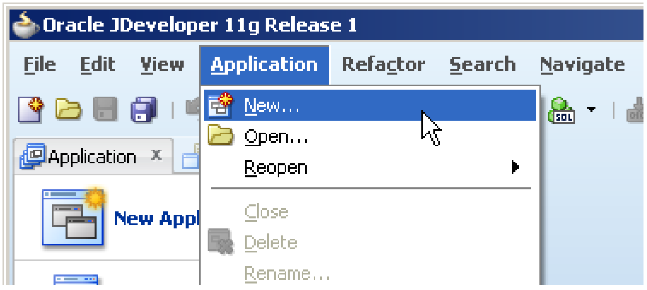
The term "Router" is used here for convenience rather than as a descriptive term. The SOA Composite which is called the "Router" will "route" each message it receives to a file in the file system. In subsequent solutions, which are developed in subsequent articles, this name becomes descriptive.

The “Router” composite will receive HL7 v2 delimited messages and will write them “as-is” to files. We made sure that messages are not translated by the ADT Receiver endpoint by unchecking the “Translation” checkbox in the endpoint configuration. As SOA Suite adapters by default deal with XML messages, we need to make sure we override default configuration of the ADT Receiver endpoint and for the File adapter which will do the writing.

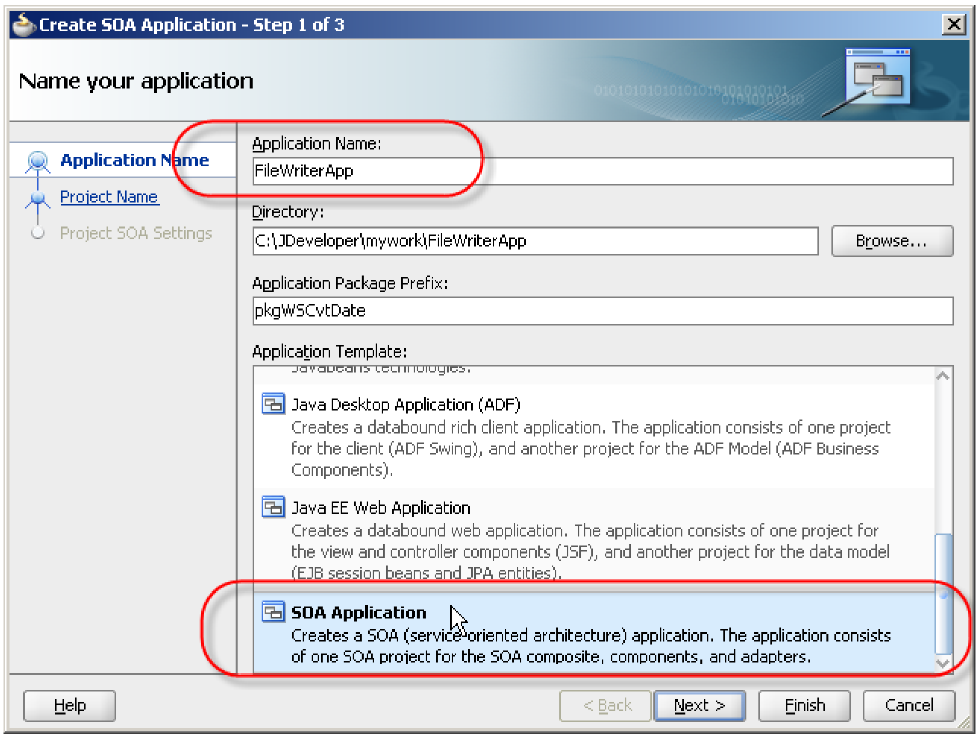
Tying together two adapters in SOA Suite requires a logic component - BPEL, Mediator, Business Rules, … . The role of this component is to receive a message from the inbound adapter and pass it to the outbound adapter, potentially transforming it in the process. At this point in the solution development 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 “FileWriterApp” as “Application Name”, choose “SOA Application” and click “Next”


Create SOA Application - Step 1 of 3

Name your application

Application Name: FileWriterApp

Directory: C:\JDeveloper\mywork\FileWriterApp

Application Package Prefix: pkgWSCvtDate

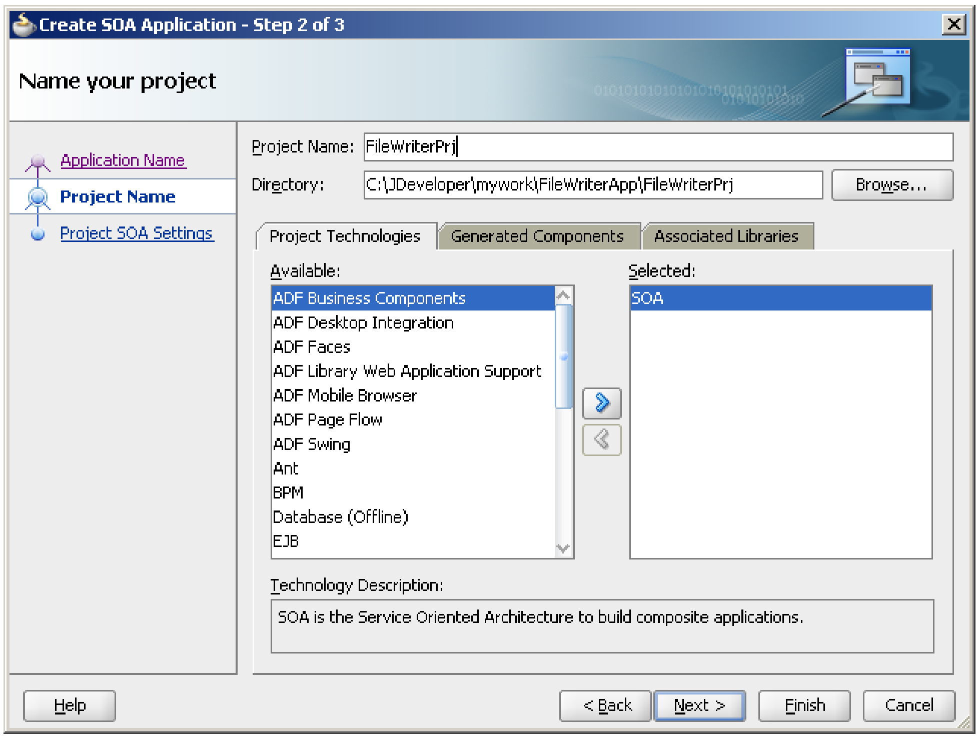
Application Template:

Java Desktop Application (ADF)
Creates a databound rich client application. The application consists of one project for the client (ADF Swing), and another project for the ADF Model (ADF Business Components).

Java EE Web Application
Creates a databound web application. The application consists of one project for the view and controller components (JSF), and another project for the data model (EJB session beans and JPA entities).

SOA Application
Creates a SOA (service-oriented architecture) application. The application consists of one SOA project for the SOA composite, components, and adapters.

☐ Enter “FileWriterPrj” as “Project Name” and click “Next”


Create SOA Application - Step 2 of 3

Name your project

Project Name: FileWriterPrj

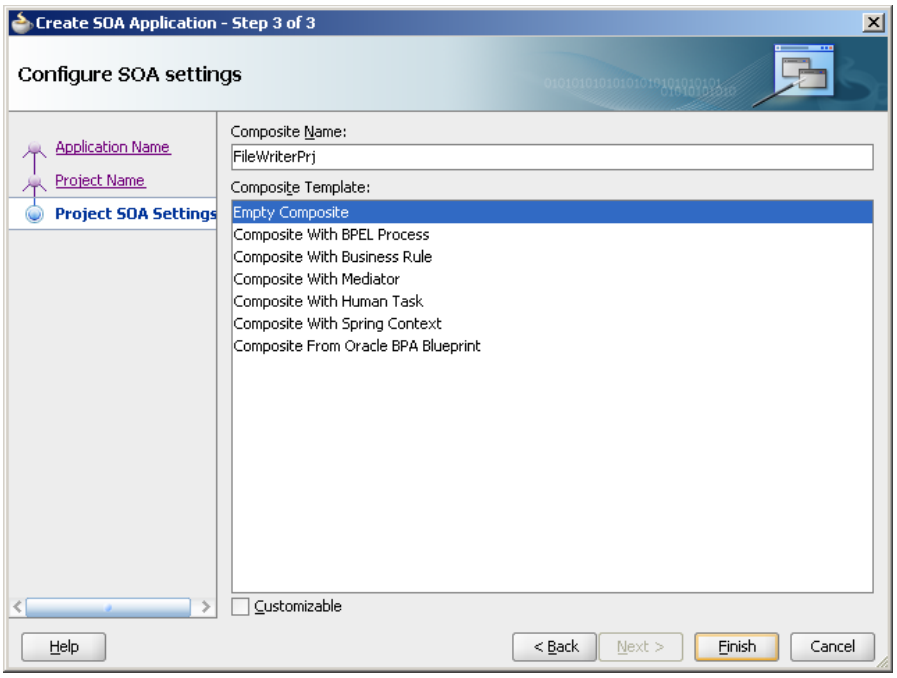
Directory: C:\JDeveloper\mywork\FileWriterApp\FileWriterPrj

Available:
ADF Business Components
ADF Desktop Integration
ADF Faces
ADF Library Web Application Support
ADF Mobile Browser
ADF Page Flow
ADF Swing
Ant
BPM
Database (Offline)
EJB

Selected:
SOA

Technology Description:
SOA is the Service Oriented Architecture to build composite applications.

☐ Accept “Empty Composite” and click “Finish”


Create SOA Application - Step 3 of 3

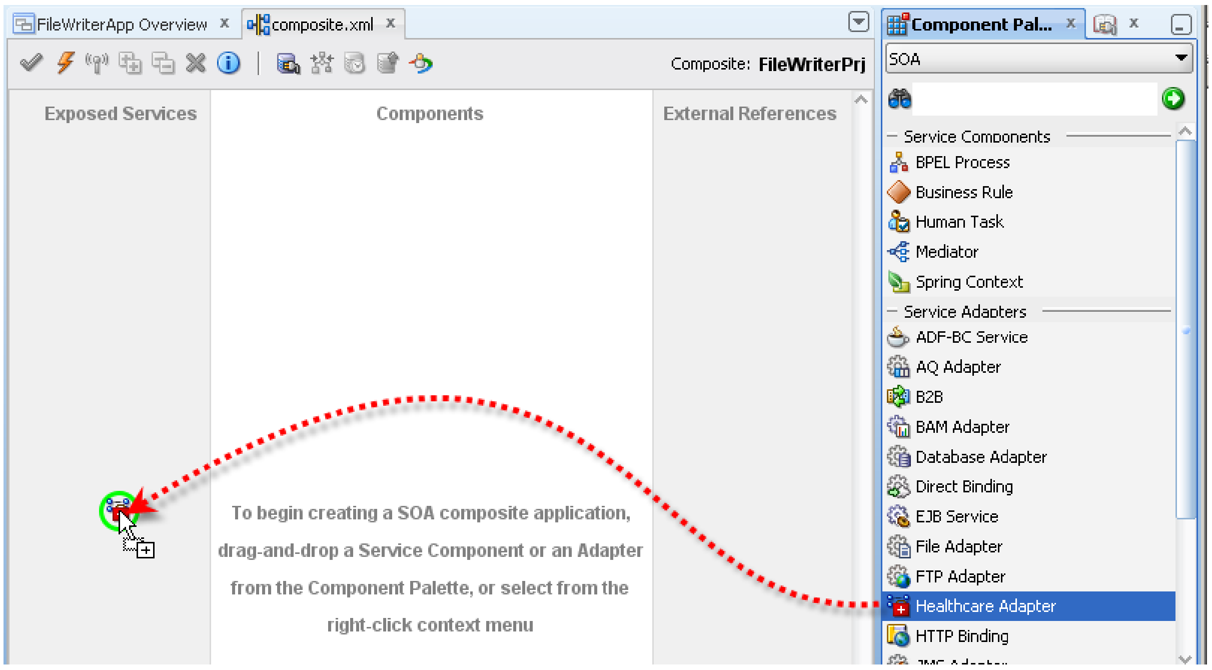
Configure SOA settings

Composite Name: FileWriterPrj

Composite Template:
Empty Composite
Composite With BPEL Process
Composite With Business Rule
Composite With Mediator
Composite With Human Task
Composite With Spring Context
Composite From Oracle BPA Blueprint

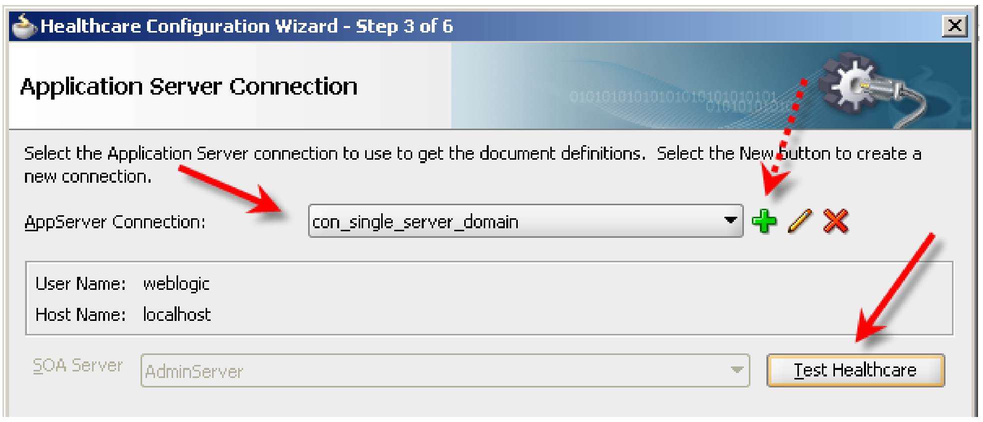
☐ Customizable

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



- ☐ Name the adapter "HCI_CMM_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"


Healthcare Configuration Wizard - Step 3 of 6

Application Server Connection

Select the Application Server connection to use to get the document definitions. Select the New button to create a new connection.

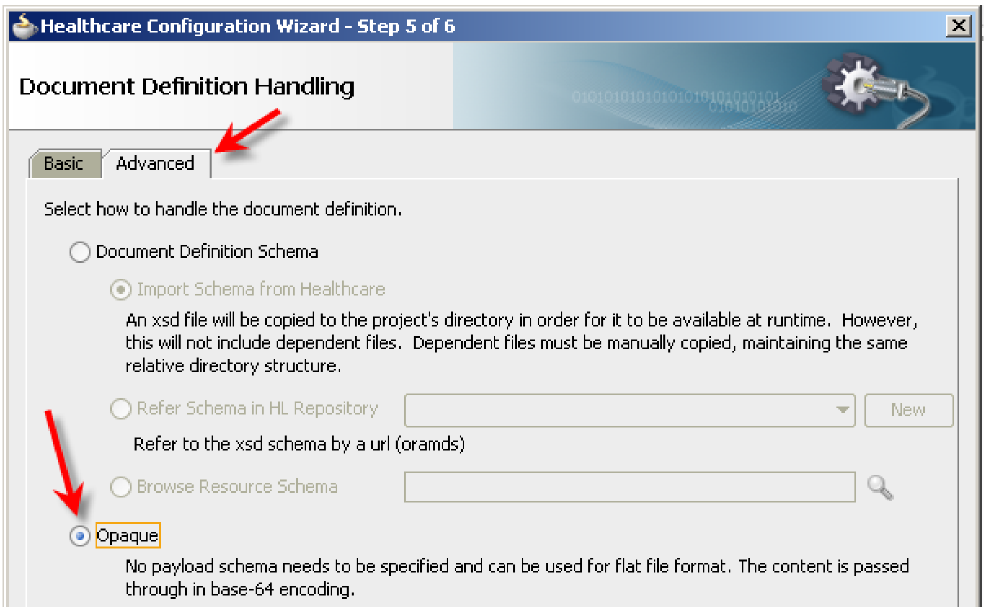
AppServer Connection: con_single_server_domain

User Name: weblogic
Host Name: localhost

SOA Server: AdminServer

- ☐ 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


Healthcare Configuration Wizard - Step 5 of 6
Document Definition Handling
Basic
Advanced

Select how to handle the document definition.

( ) Document Definition Schema

(•) Import Schema from Healthcare

An xsd file will be copied to the project's directory in order for it to be available at runtime. However, this will not include dependent files. Dependent files must be manually copied, maintaining the same relative directory structure.

( ) Refer Schema in HL Repository

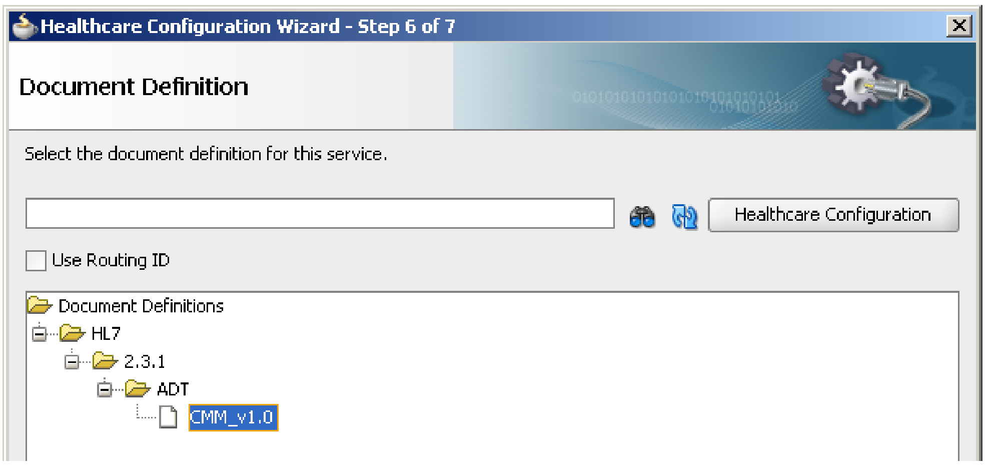
Refer to the xsd schema by a url (oramds)

( ) Browse Resource Schema

[✓ Opaque]

No payload schema needs to be specified and can be used for flat file format. The content is passed through in base-64 encoding.

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



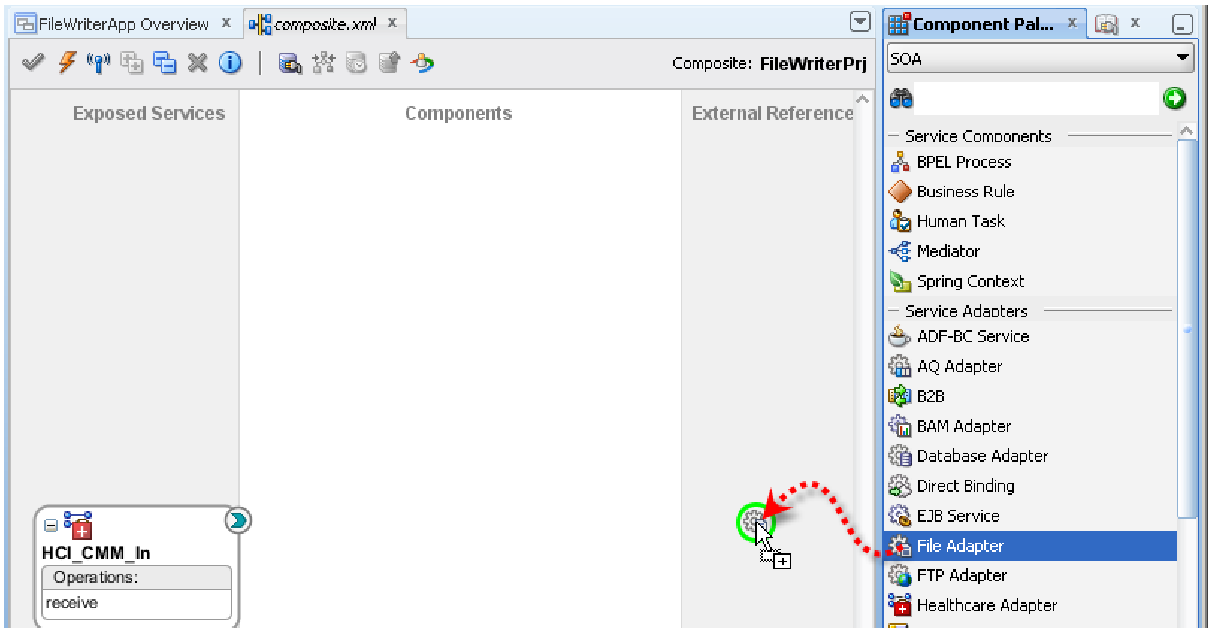
Select the document definition for this service.

☐ Use Routing ID

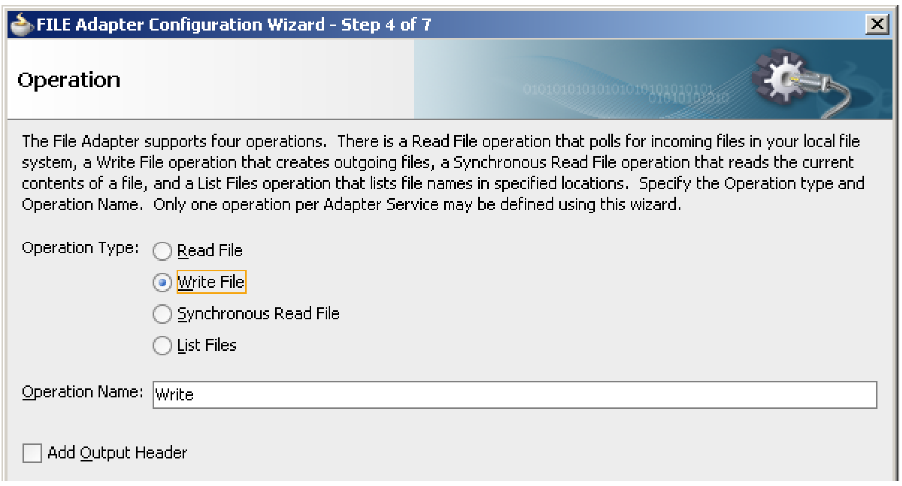
The Healthcare Adapter is configured to receive untranslated HL7 messages from the ADT Receiver endpoint. The two work in concert – one receives and acknowledges messages and the other makes them available for processing by a SOA Composite.

Let's now add and configure the File Adapter.

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



- ☐ Name it "File_CMM_Out" and click "Next"
- ☐ Accept the default "Define from operation and schema (defined later)" "Adapter Interface" and click "Next"
- ☐ Choose "Write File" in the "Operation" dialogue box and click "Next"


FILE Adapter Configuration Wizard - Step 4 of 7

Operation

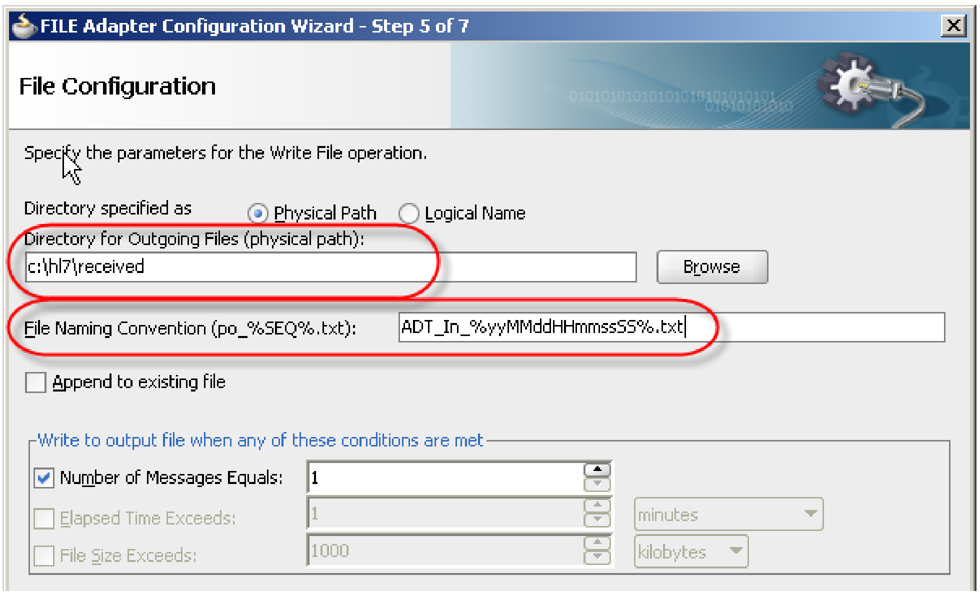
The File Adapter supports four operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, a Synchronous Read File operation that reads the current contents of a file, and a List Files operation that lists file names in specified locations. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type: ○ Read File [✓ Write File] ○ Synchronous Read File ○ List Files

Operation Name: Write

☐ Add Output Header

- ☐ Specify the path to which to write the file and a file name of the form "ADT_In_%yyMMddHHmmssSS%.txt", then click "Next" – note that the file name for each message will be different and will embed a timestamp to tell message files apart


FILE Adapter Configuration Wizard - Step 5 of 7

File Configuration

Specify the parameters for the Write File operation.

Directory specified as [✓ Physical Path] Logical Name

Directory for Outgoing Files (physical path): c:\hl7\received

File Naming Convention (po_%SEQ%.txt): ADT_In_%yyMMddHHmmssSS%.txt

☐ Append to existing file

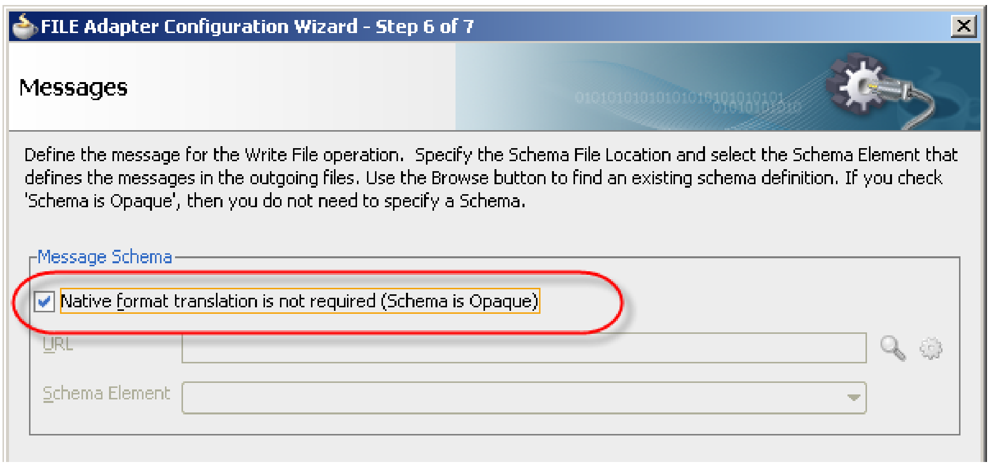
Write to output file when any of these conditions are met

☑ Number of Messages Equals: 1

☐ Elapsed Time Exceeds: 1 minutes

☐ File Size Exceeds: 1000 kilobytes

☐ Check the "Native format translation is not required (Schema is Opaque)" checkbox, click "Next" and "Finish" – note that here, too, we are configuring the adapter to treat the message as an opaque message


FILE Adapter Configuration Wizard - Step 6 of 7

Messages

Define the message for the Write File operation. Specify the Schema File Location and select the Schema Element that defines the messages in the outgoing files. Use the Browse button to find an existing schema definition. If you check 'Schema is Opaque', then you do not need to specify a Schema.

Message Schema

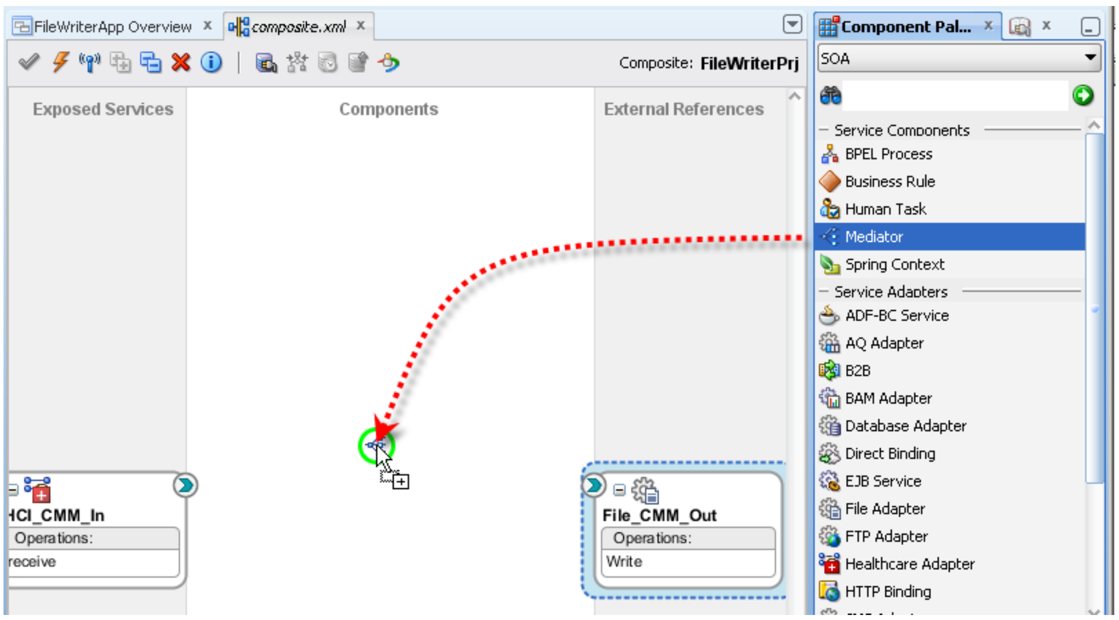
☑ Native format translation is not required (Schema is Opaque)

URL

Schema Element

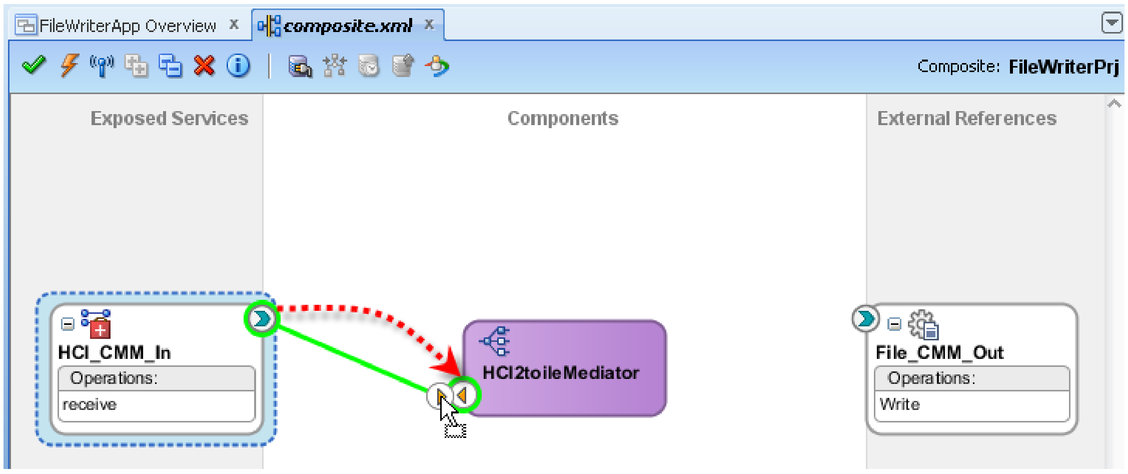
The File adapter is configured. We now need to tie the inbound and the outbound with a "Service Component". We will use "Mediator" component for simplicity.

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

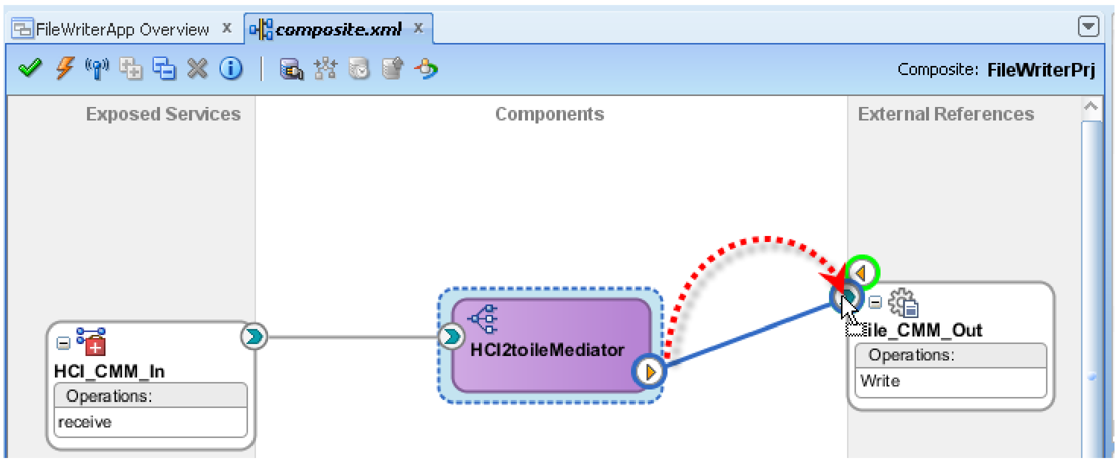


☐ Name the component “HCI2toileMediator” and click “OK”

☐ Drag from the “Chevron in a circle” symbol in the top right hand corner of the HCI_CMM_In adapter to the HCItoFileMediator component’s left pointing triangle to connect the two components



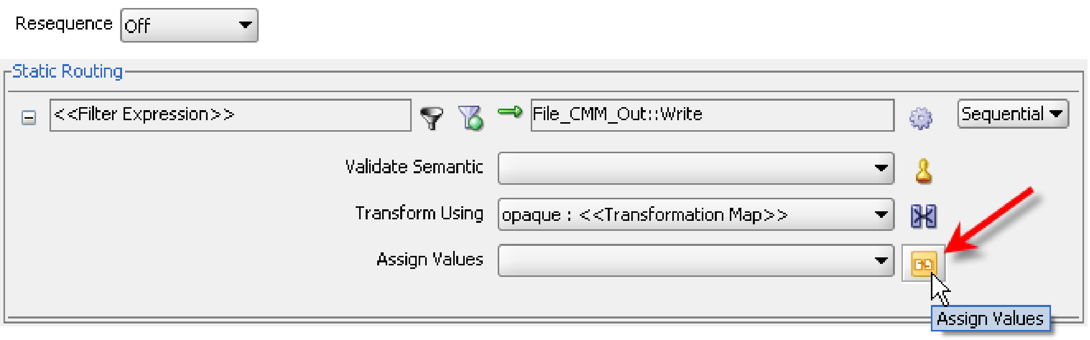
☐ Drag from the right hand pointing triangle in the HCItoFileMediator to the chevron in a circle symbol in the top left corner of the File_CMM_Out adapter to connect the two



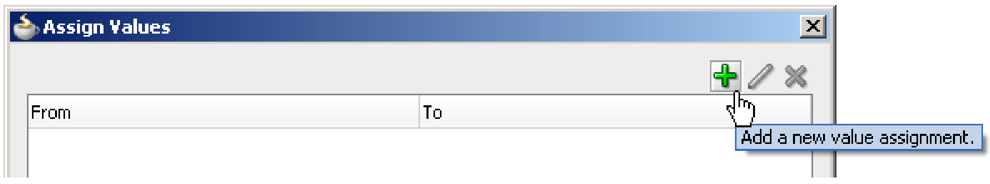
If we were passing through XML messages we would be done configuring the Mediator component. By default it will simply pass its input to its output. Because we are using opaque messages we must explicitly configure this pass through.

☐ Double-click the HCItoFileMediator component to open its properties

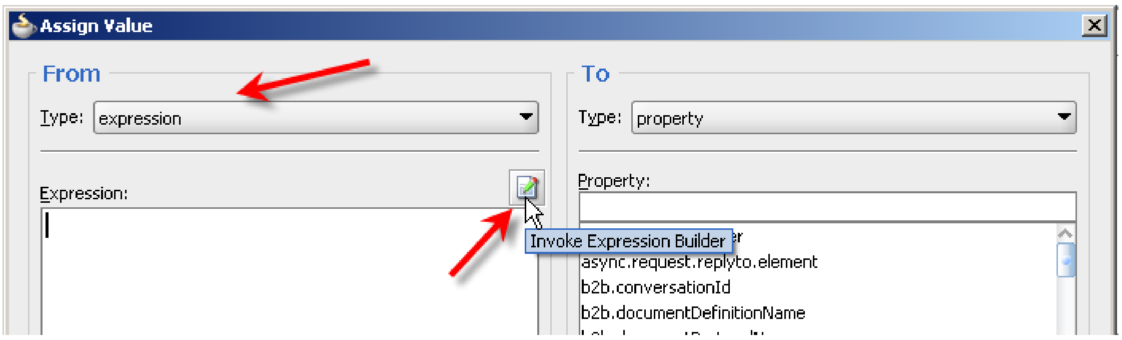
☐ Click on the "Assign Values" button to start assigning values – we onlt need to assign the input to the output



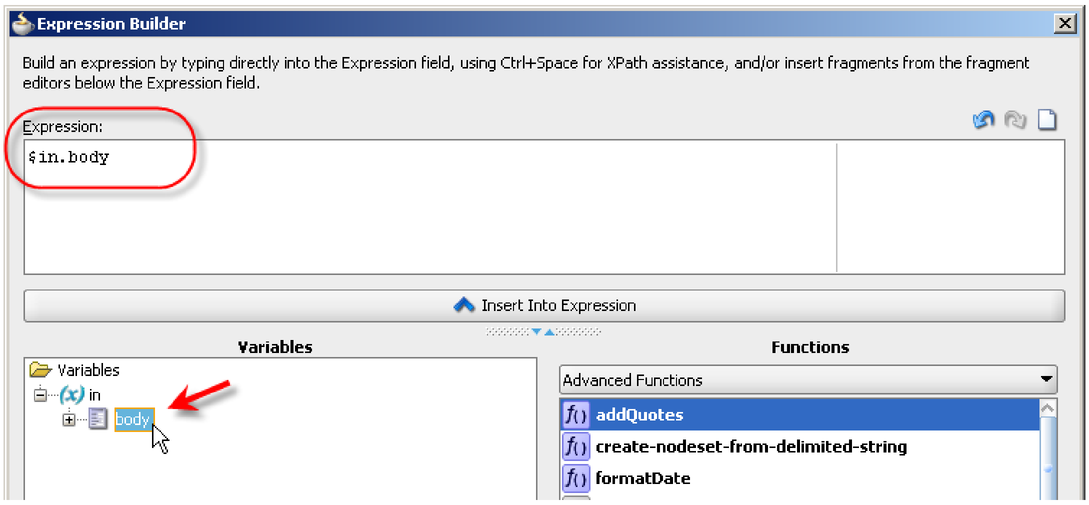
☐ Click the "Add a new value assignment" (the plus sign) button



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



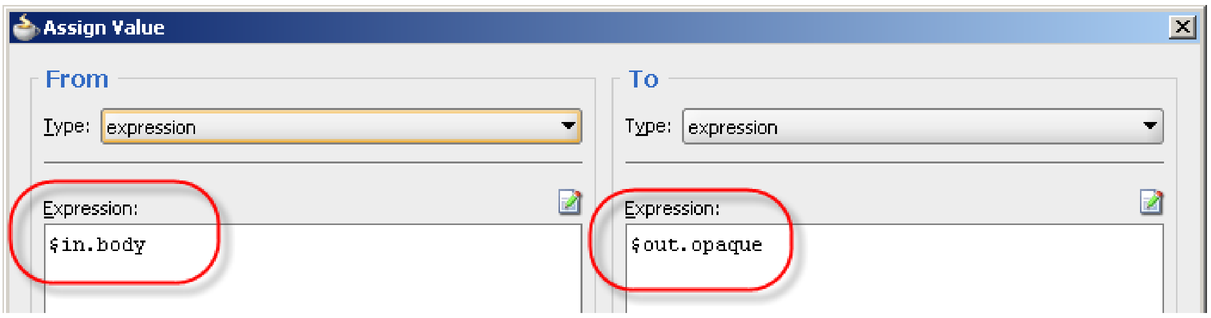
- ☐ Expand the "in" variable and double-click the "body" node to make it appear in the "Expression" box – alternatively select the "body" node and click the "Insert Into Expression" button – then click "OK"


Build an expression by typing directly into the Expression field, using Ctrl+Space for XPath assistance, and/or insert fragments from the fragment editors below the Expression field.

Expression:

```
$in.body
```

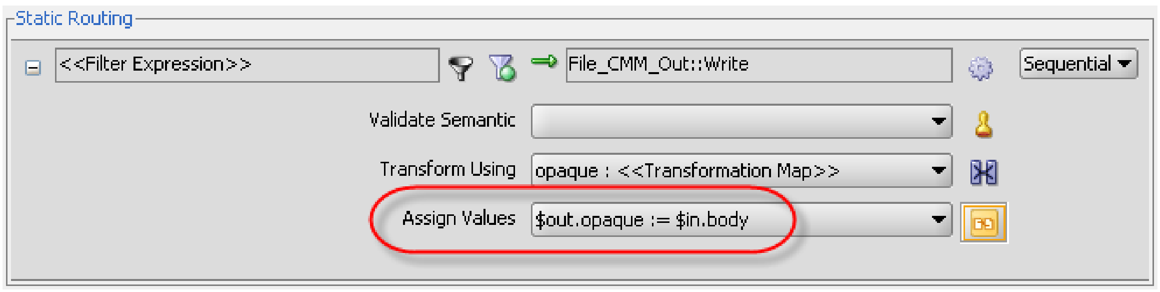
- ☐ Repeat the process for the "To" side, choosing "expression" for "Type", clicking the "Invoke Expression Builder" and adding the $out.opaque expression


From — Type: expression — Expression: `$in.body`

To — Type: expression — Expression: `$out.opaque`

- ☐ Click "OK" and "OK" again to complete assignment

The Mediator properties we changed will look like the illustration below


Static Routing

<<Filter Expression>> File_CMM_Out::Write Sequential

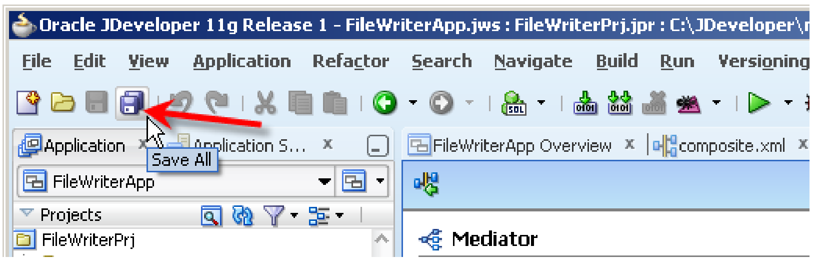
Validate Semantic

Transform Using opaque : <<Transformation Map>>

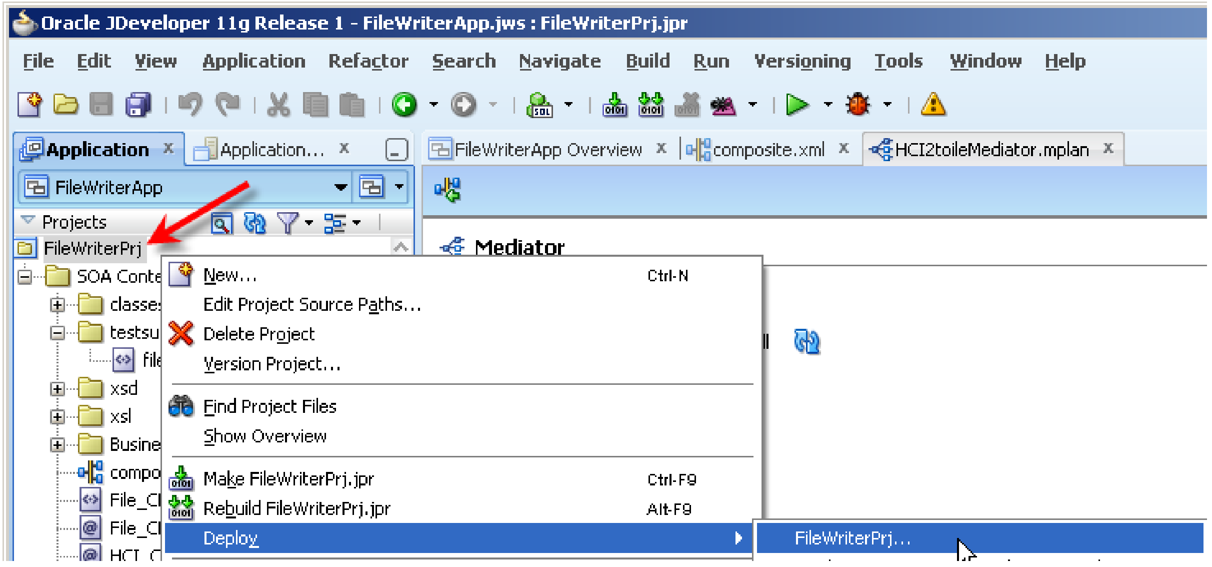
Assign Values $out.opaque := $in.body

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

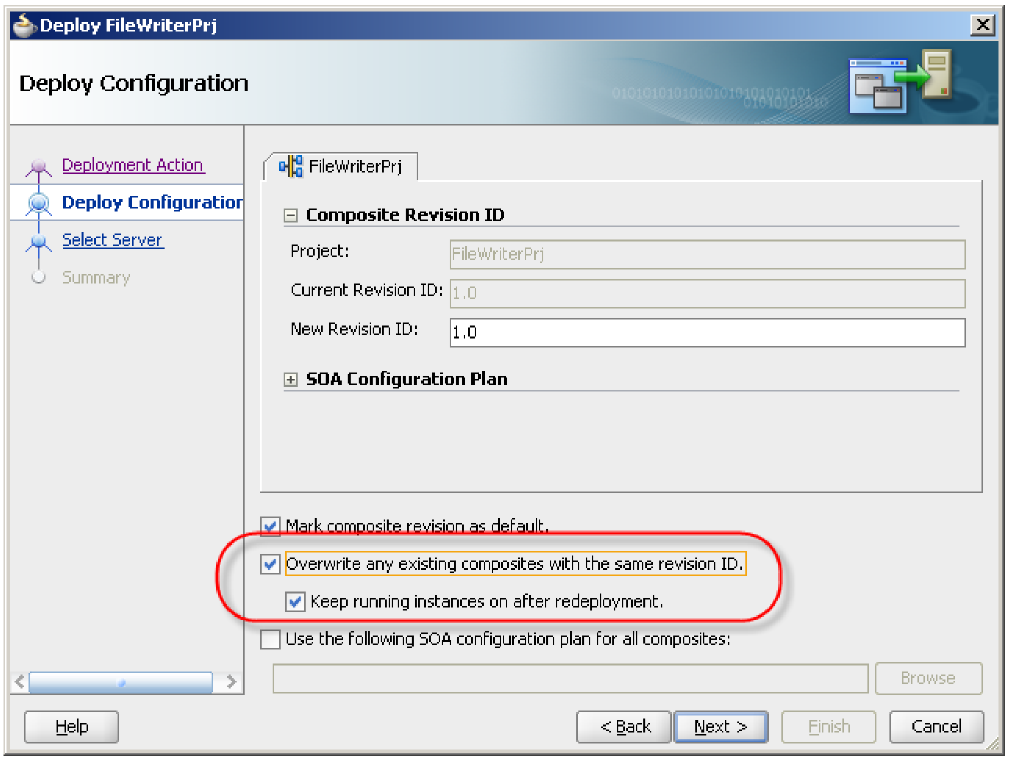

Oracle JDeveloper 11g Release 1 - FileWriterApp.jws : FileWriterPrj.jpr : C:\JDeveloper\
File Edit View Application Refactor Search Navigate Build Run Versioning
Application
Save All
FileWriterApp
Projects
FileWriterPrj
FileWriterApp Overview
composite.xml
Mediator

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


Oracle JDeveloper 11g Release 1 - FileWriterApp.jws : FileWriterPrj.jpr
File Edit View Application Refactor Search Navigate Build Run Versioning Tools Window Help
Application
Application…
FileWriterApp Overview
composite.xml
HCI2toileMediator.mplan
FileWriterApp
Projects
FileWriterPrj
Mediator
New… Ctrl-N
Edit Project Source Paths…
Delete Project
Version Project…
Find Project Files
Show Overview
Make FileWriterPrj.jpr Ctrl-F9
Rebuild FileWriterPrj.jpr Alt-F9
Deploy
FileWriterPrj…

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

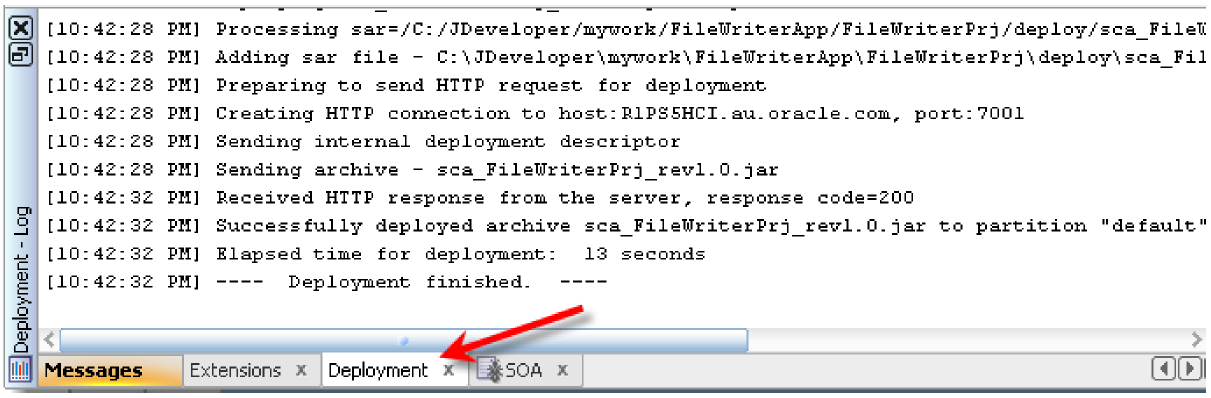
☐ Check the "Overwrite any existing composite with the same revision ID." Checkobox and click "Next"



Project: FileWriterPrj
Current Revision ID: 1.0
New Revision ID: 1.0

☑ Mark composite revision as default.
☑ Overwrite any existing composites with the same revision ID.
☑ Keep running instances on after redeployment.
☐ Use the following SOA configuration plan for all composites:

- ☐ Choose the correct application server connection and click "Finish"
- ☐ Observe deployment log messages looking for completion without errors



```
[10:42:28 PM] Processing sar=/C:/JDeveloper/mywork/FileWriterApp/FileWriterPrj/deploy/sca_FileW
[10:42:28 PM] Adding sar file - C:\JDeveloper\mywork\FileWriterApp\FileWriterPrj\deploy\sca_Fil
[10:42:28 PM] Preparing to send HTTP request for deployment
[10:42:28 PM] Creating HTTP connection to host:R1PS5HCI.au.oracle.com, port:7001
[10:42:28 PM] Sending internal deployment descriptor
[10:42:28 PM] Sending archive - sca_FileWriterPrj_rev1.0.jar
[10:42:32 PM] Received HTTP response from the server, response code=200
[10:42:32 PM] Successfully deployed archive sca_FileWriterPrj_rev1.0.jar to partition "default"
[10:42:32 PM] Elapsed time for deployment:  13 seconds
[10:42:32 PM] ----  Deployment finished.  ----
```

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.

Process ADT messages

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 output in our configured output directory – for me c:\hl7\received, 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 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.

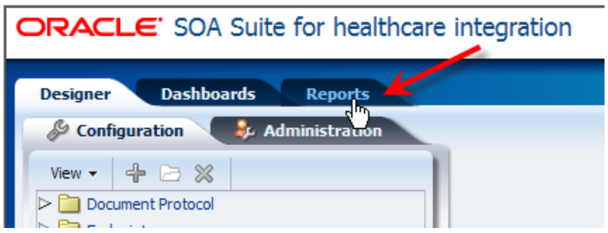
Explore Message Tracking

Let's explore message tracking.

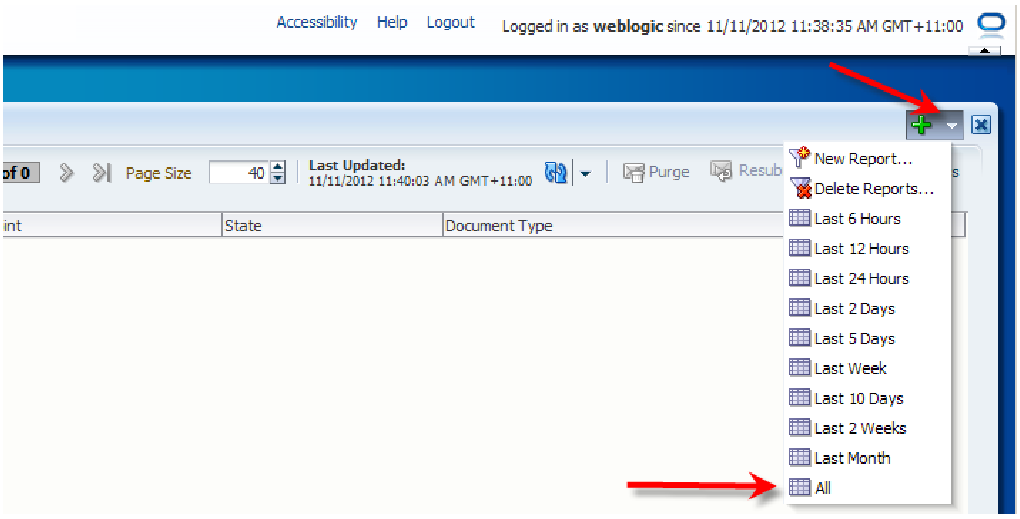
- ☐ Start the Healthcare Integration Console – http://localhost:7001/healthcare

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

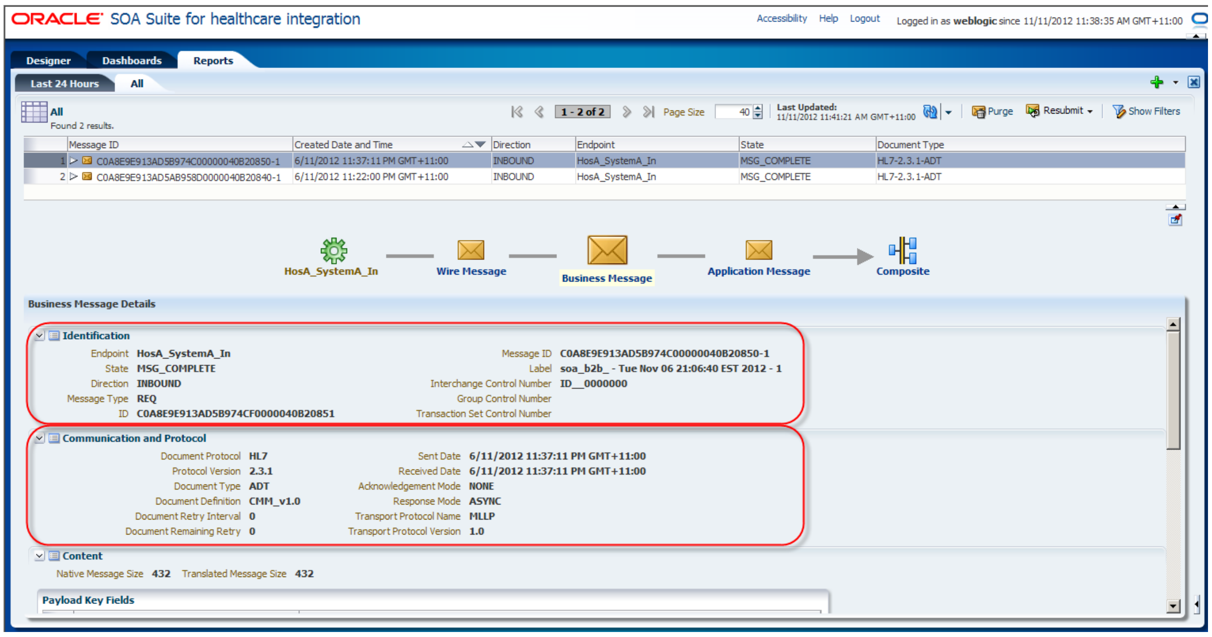
- ☐ Click the "Reports" Tab



- ☐ If you submitted two messages following instructions in the previous section, but you don't see any messages here, and you are looking at this text more than 24 hours after you did the previous section, then pull down the "Plus" sign drop down and choose "All"



- ☐ Select the first message in the list (there ought to be only 2 – or select the second last if there are more than two because you submitted more than two), 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



All
Found 2 results.

	Message ID	Created Date and Time	Direction	Endpoint	State	Document Type
1	C0A8E9E913AD5B974C00000040B20850-1	6/11/2012 11:37:11 PM GMT+11:00	INBOUND	HosA_SystemA_In	MSG_COMPLETE	HL7-2.3.1-ADT
2	C0A8E9E913AD5AB958D0000040B20840-1	6/11/2012 11:22:00 PM GMT+11:00	INBOUND	HosA_SystemA_In	MSG_COMPLETE	HL7-2.3.1-ADT

Business Message Details

Identification

Endpoint	HosA_SystemA_In	Message ID	C0A8E9E913AD5B974C00000040B20850-1
State	MSG_COMPLETE	Label	soa_b2b_ - Tue Nov 06 21:06:40 EST 2012 - 1
Direction	INBOUND	Interchange Control Number	ID__0000000
Message Type	REQ	Group Control Number	
ID	C0A8E9E913AD5B974CF0000040B20851	Transaction Set Control Number	

Communication and Protocol

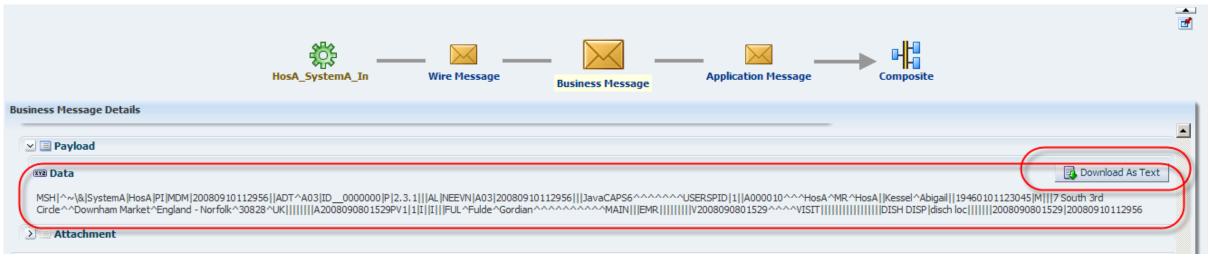
Document Protocol	HL7	Sent Date	6/11/2012 11:37:11 PM GMT+11:00
Protocol Version	2.3.1	Received Date	6/11/2012 11:37:11 PM GMT+11:00
Document Type	ADT	Acknowledgement Mode	NONE
Document Definition	CMM_v1.0	Response Mode	ASYNC
Document Retry Interval	0	Transport Protocol Name	MLLP
Document Remaining Retry	0	Transport Protocol Version	1.0

Content

Native Message Size 432 Translated Message Size 432

Payload Key Fields

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



Business Message Details

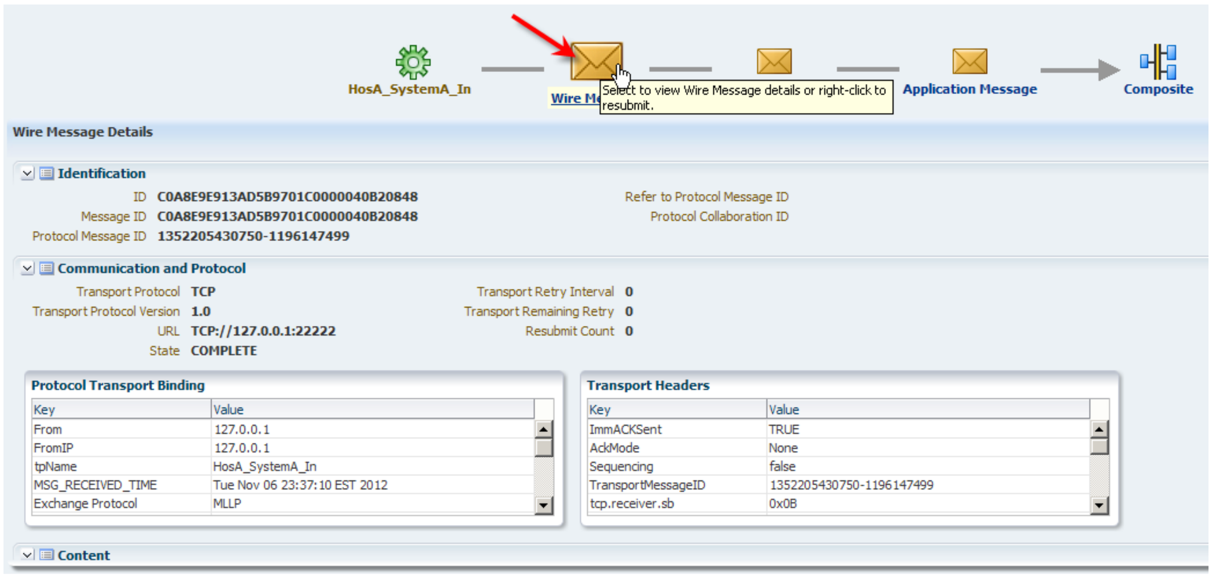
Payload

Data

```
MSH|^~\&|SystemA|HosA|PI|MDM|20080910112956||ADT^A03|ID__0000000|P|2.3.1|||AL|NEEVN|A03|20080910112956|||JavaCAPS6^^^^^^^USERSPID|1||A000010^^^HosA^MR^HosA||Kessel^Abigail||19460101123045|M|||7 South 3rd
Circle^^Downham Market^England - Norfolk^30828^UK||||||||A2008090801529PV1|1|I||I|||FUL^Fulde^Gordian^^^^^^^^^^MAIN|||EMR||||||||V2008090801529^^^^VISIT||||||||||||||||DISH DISP|disch loc|||||||2008090801529|20080910112956
```

Attachment

☐ Click the "Wire Message" Icon or link, inspect the wire message-related attribute values and the message content


HosA_SystemA_In
Wire Me
Select to view Wire Message details or right-click to resubmit.
Application Message
Composite

Wire Message Details

Identification

ID	C0A8E9E913AD5B9701C0000040B20848	Refer to Protocol Message ID	
Message ID	C0A8E9E913AD5B9701C0000040B20848	Protocol Collaboration ID	
Protocol Message ID	1352205430750-1196147499		

Communication and Protocol

Transport Protocol	TCP	Transport Retry Interval	0
Transport Protocol Version	1.0	Transport Remaining Retry	0
URL	TCP://127.0.0.1:22222	Resubmit Count	0
State	COMPLETE		

Protocol Transport Binding

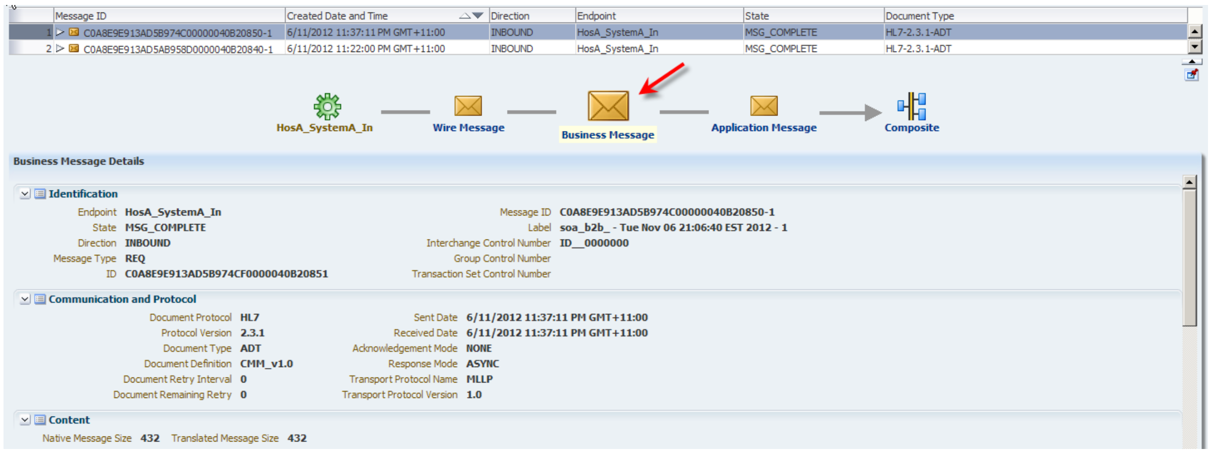
Key	Value
From	127.0.0.1
FromIP	127.0.0.1
tpName	HosA_SystemA_In
MSG_RECEIVED_TIME	Tue Nov 06 23:37:10 EST 2012
Exchange Protocol	MLLP

Transport Headers

Key	Value
ImmACKSent	TRUE
AckMode	None
Sequencing	false
TransportMessageID	1352205430750-1196147499
tcp.receiver.sb	0x0B

Content

☐ Click the "Business Message" icon/link and note that by default this is the view presented when a message us selected in the list of messages



	Message ID	Created Date and Time	Direction	Endpoint	State	Document Type
1	C0A8E9E913AD5B974C00000040B20850-1	6/11/2012 11:37:11 PM GMT+11:00	INBOUND	HosA_SystemA_In	MSG_COMPLETE	HL7-2.3.1-ADT
2	C0A8E9E913AD5AB958D0000040B20840-1	6/11/2012 11:22:00 PM GMT+11:00	INBOUND	HosA_SystemA_In	MSG_COMPLETE	HL7-2.3.1-ADT

Business Message Details

Identification

Endpoint	HosA_SystemA_In	Message ID	C0A8E9E913AD5B974C00000040B20850-1
State	MSG_COMPLETE	Label	soa_b2b_ - Tue Nov 06 21:06:40 EST 2012 - 1
Direction	INBOUND	Interchange Control Number	ID__0000000
Message Type	REQ	Group Control Number	
ID	C0A8E9E913AD5B974CF0000040B20851	Transaction Set Control Number	

Communication and Protocol

Document Protocol	HL7	Sent Date	6/11/2012 11:37:11 PM GMT+11:00
Protocol Version	2.3.1	Received Date	6/11/2012 11:37:11 PM GMT+11:00
Document Type	ADT	Acknowledgement Mode	NONE
Document Definition	CMM_v1.0	Response Mode	ASYNC
Document Retry Interval	0	Transport Protocol Name	MLLP
Document Remaining Retry	0	Transport Protocol Version	1.0

Content

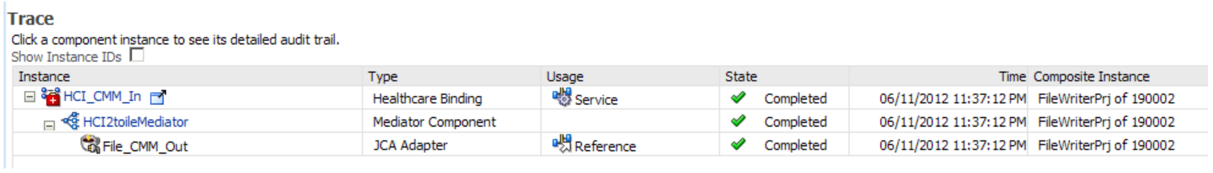
Native Message Size 432 Translated Message Size 432

☐ 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 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".

☐ 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



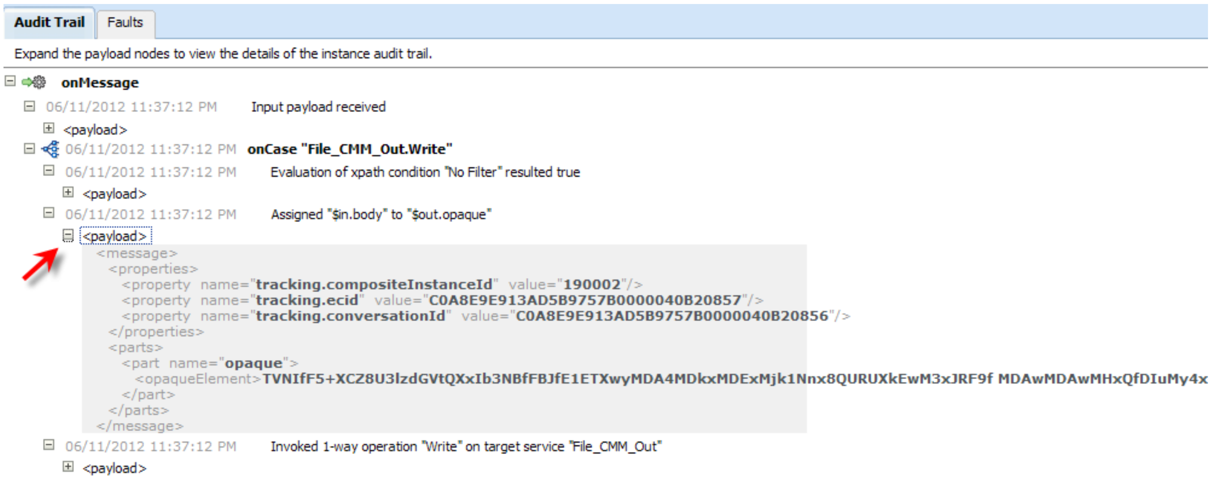
Trace

Click a component instance to see its detailed audit trail.

Show Instance IDs ☐

Instance	Type	Usage	State	Time	Composite Instance
⊟ HCI_CMM_In	Healthcare Binding	Service	✓ Completed	06/11/2012 11:37:12 PM	FileWriterPrj of 190002
⊟ HCI2toileMediator	Mediator Component		✓ Completed	06/11/2012 11:37:12 PM	FileWriterPrj of 190002
File_CMM_Out	JCA Adapter	Reference	✓ Completed	06/11/2012 11:37:12 PM	FileWriterPrj of 190002

☐ Click the "HCI2FileMediator" 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



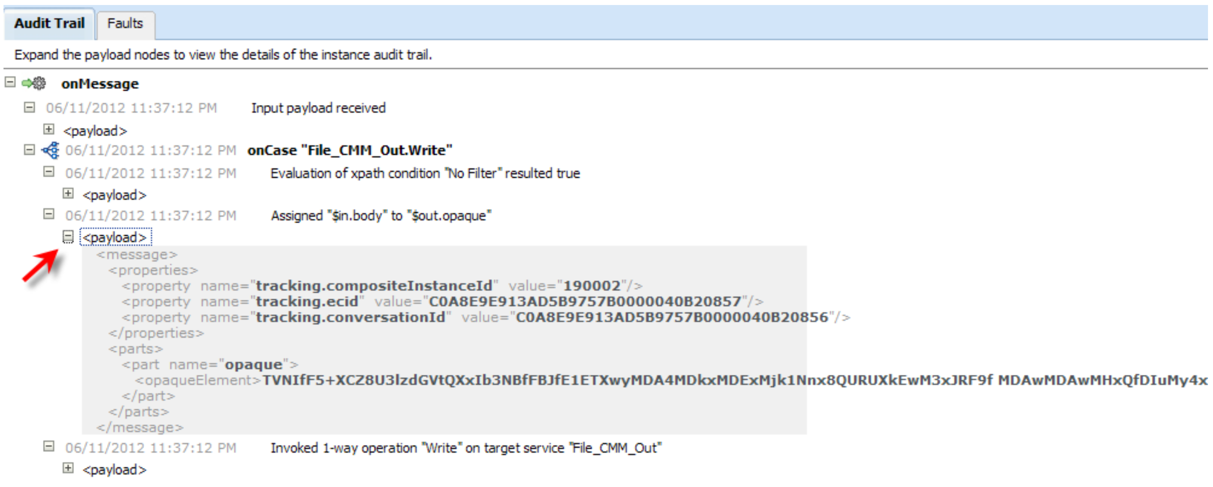
Expand the payload nodes to view the details of the instance audit trail.

```
⊟ onMessage
  ⊟ 06/11/2012 11:37:12 PM      Input payload received
     ⊞ <payload>
  ⊟ 06/11/2012 11:37:12 PM  onCase "File_CMM_Out.Write"
     ⊟ 06/11/2012 11:37:12 PM      Evaluation of xpath condition "No Filter" resulted true
        ⊞ <payload>
     ⊟ 06/11/2012 11:37:12 PM      Assigned "$in.body" to "$out.opaque"
        ⊟ <payload>
```

```
<message>
  <properties>
    <property  name="tracking.compositeInstanceId"  value="190002"/>
    <property  name="tracking.ecid"  value="C0A8E9E913AD5B9757B0000040B20857"/>
    <property  name="tracking.conversationId"  value="C0A8E9E913AD5B9757B0000040B20856"/>
  </properties>
  <parts>
    <part  name="opaque">
      <opaqueElement>TVNIfF5+XCZ8U3lzdGVtQXxIb3NBfFBJfE1ETXwyMDA4MDkxMDExMjk1Nnx8QURUXkEwM3xJRF9f MDAwMDAwMHxQfDIuMy4x
    </part>
  </parts>
</message>
```

```
     ⊟ 06/11/2012 11:37:12 PM      Invoked 1-way operation "Write" on target service "File_CMM_Out"
        ⊞ <payload>
```

☐ Note the property values and message content – these will be much more "interesting" when we perform message translation in the next section – close the Enterprise Manager windows



Expand the payload nodes to view the details of the instance audit trail.

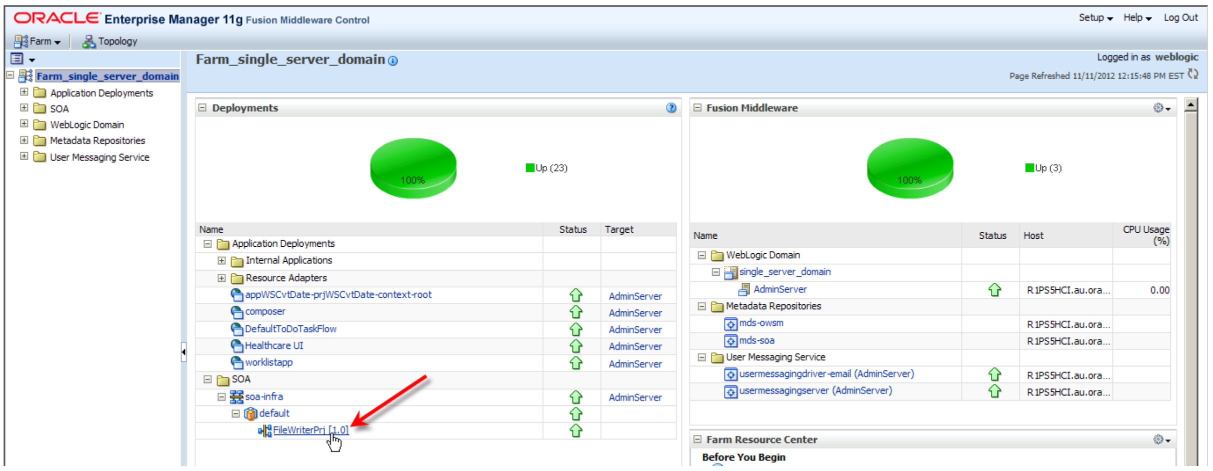
```
⊟ onMessage
  ⊟ 06/11/2012 11:37:12 PM    Input payload received
      ⊞ <payload>
  ⊟ 06/11/2012 11:37:12 PM  onCase "File_CMM_Out.Write"
      ⊟ 06/11/2012 11:37:12 PM    Evaluation of xpath condition "No Filter" resulted true
          ⊞ <payload>
      ⊟ 06/11/2012 11:37:12 PM    Assigned "$in.body" to "$out.opaque"
          ⊟ <payload>
```

```
<message>
  <properties>
    <property name="tracking.compositeInstanceId" value="190002"/>
    <property name="tracking.ecid" value="C0A8E9E913AD5B9757B0000040B20857"/>
    <property name="tracking.conversationId" value="C0A8E9E913AD5B9757B0000040B20856"/>
  </properties>
  <parts>
    <part name="opaque">
      <opaqueElement>TVNIfF5+XCZ8U3lzdGVtQXxIb3NBfFBJfE1ETXwyMDA4MDkxMDExMjk1Nnx8QURUXkEwM3xJRF9f MDAwMDAwMHxQfDIuMy4x
    </part>
  </parts>
</message>
```

```
      ⊟ 06/11/2012 11:37:12 PM    Invoked 1-way operation "Write" on target service "File_CMM_Out"
          ⊞ <payload>
```

Note that the Base64-encoded HL7 v2 Delimited message is the content of the opaqueElements node in the payload XML structure

☐ Explicitly open the Enterprise Manager Console - http://localhost:7001/em - and click on the name of the FileWriterPrj composite

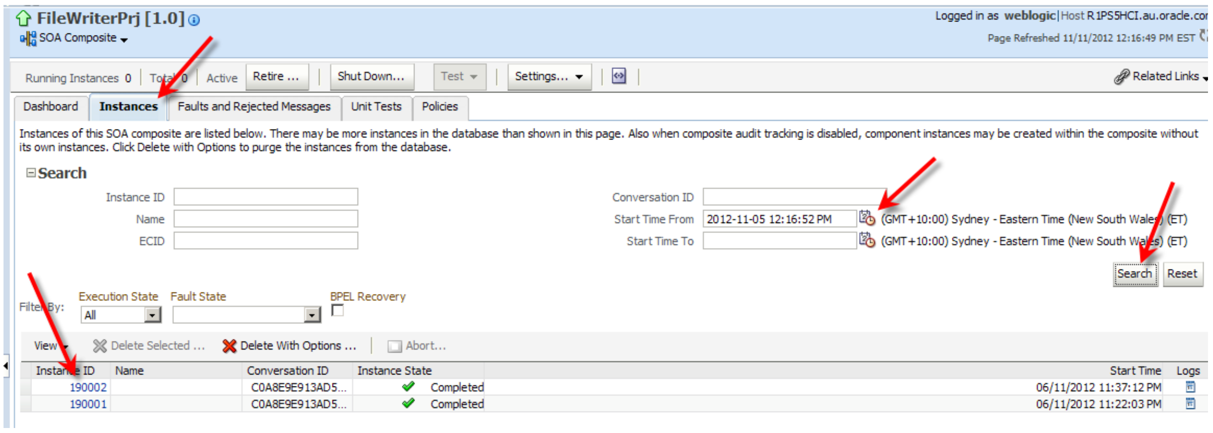


Farm_single_server_domain

Name	Status	Target
⊟ Application Deployments		
⊞ Internal Applications		
⊞ Resource Adapters		
appWSCvtDate-prjWSCvtDate-context-root	⇧	AdminServer
composer	⇧	AdminServer
DefaultToDoTaskFlow	⇧	AdminServer
Healthcare UI	⇧	AdminServer
worklistapp	⇧	AdminServer
⊟ SOA		
⊟ soa-infra	⇧	AdminServer
⊟ default	⇧	
FileWriterPrj [1.0]	⇧	

Name	Status	Host	CPU Usage (%)
⊟ WebLogic Domain			
⊟ single_server_domain			
AdminServer	⇧	R1PS5HCI.au.ora...	0.00
⊟ Metadata Repositories			
mds-owsm		R1PS5HCI.au.ora...	
mds-soa		R1PS5HCI.au.ora...	
⊟ User Messaging Service			
usermessagingdriver-email (AdminServer)	⇧	R1PS5HCI.au.ora...	
usermessagingserver (AdminServer)	⇧	R1PS5HCI.au.ora...	

☐ Click the "Instances" Tab, choose the starting date range to include the period during which instances were executed, click the "Search" button and click on the "Instance ID" link for one of the instances – not that this brings the same display as that shown when the "Instance" link was clicked in the message tracking window of the "Healthcare Integration Console"



Instances of this SOA composite are listed below. There may be more instances in the database than shown in this page. Also when composite audit tracking is disabled, component instances may be created within the composite without its own instances. Click Delete with Options to purge the instances from the database.

Instance ID	
Name	
ECID	
Conversation ID	
Start Time From	2012-11-05 12:16:52 PM (GMT+10:00) Sydney - Eastern Time (New South Wales) (ET)
Start Time To	(GMT+10:00) Sydney - Eastern Time (New South Wales) (ET)

Filter By: Execution State: All; Fault State: ; BPEL Recovery: ☐

Instance ID	Name	Conversation ID	Instance State	Start Time	Logs
190002		C0A8E9E913AD5...	✓ Completed	06/11/2012 11:37:12 PM	
190001		C0A8E9E913AD5...	✓ Completed	06/11/2012 11:22:03 PM	

☐ Close the Enterprise Manager Console windows

HL7 v2 Inbound to file with XML translation

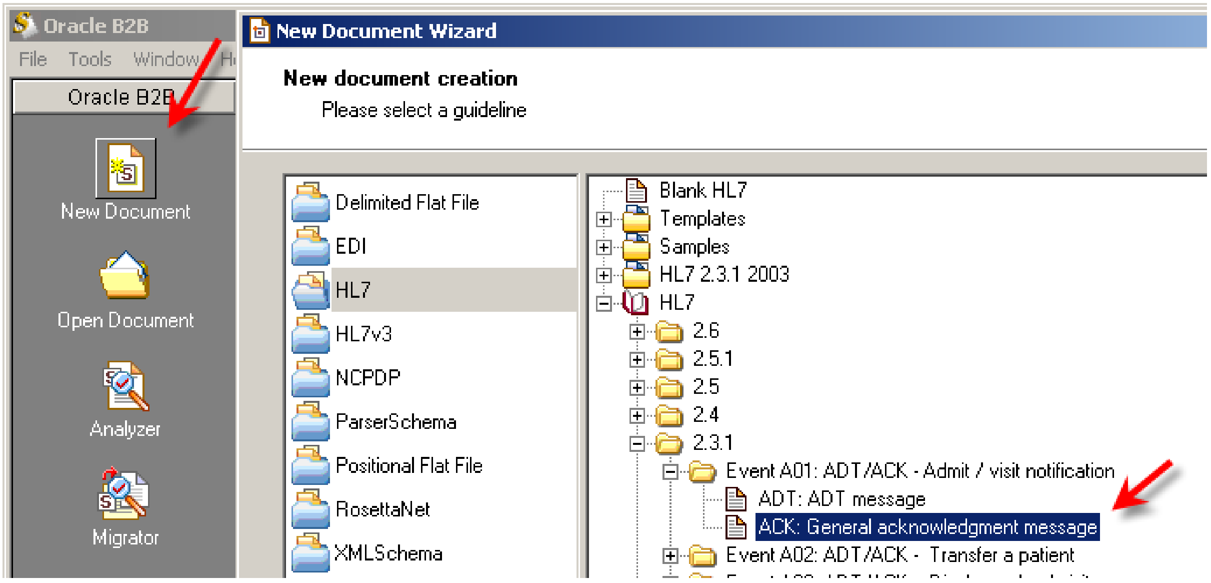
In the previous section we received a HL7 v2 delimited message and wrote it as-is to a file in the file system. In this section we will re-configure the receiving endpoint to cause the message to be automatically translated to XML. We will also re-configure acknowledgement processing so that the Functional Acknowledgement is sent after the message is parsed and translated, and that it correctly indicates the outcome of translation and validation, if any.

Define Functional Acknowledgement Document

In the case of the “Immediate ACK” the infrastructure implicitly generated the acknowledgement. In the case where the Fictional Acknowledgement is used, the case we are working through now, we must explicitly create and “introduce” the ACK message structure so it can be configured as outbound document the endpoint will be sending.

☐ Start the Oracle Document Editor

☐ Click the “New Document” button, expand the “HL7”→”2.3.1”→”Event A01 : …”, select the “ACK: General acknowledgement message” node and click “Next”



☐ Click the “Save” button and save the ECS file with the name of “ACK_2.3.1.ecs”

☐ Pull down the “File” men, select the “Export …” option, select the “Oracle B2B 2.0” option and click “Next”

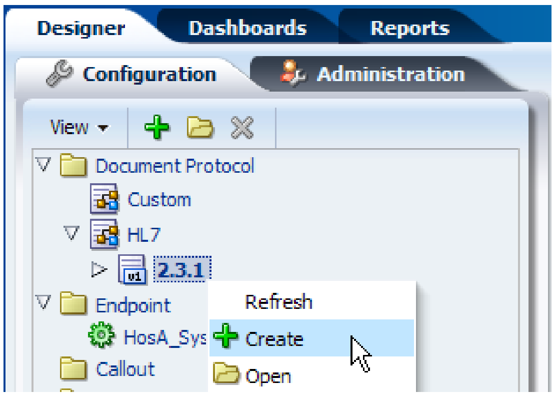
☐ Save the XSD file as ACK_2.3.1.xsd in the same location as the corresponding ECS file and click “Finish”

☐ Exit the Oracle Document Editor

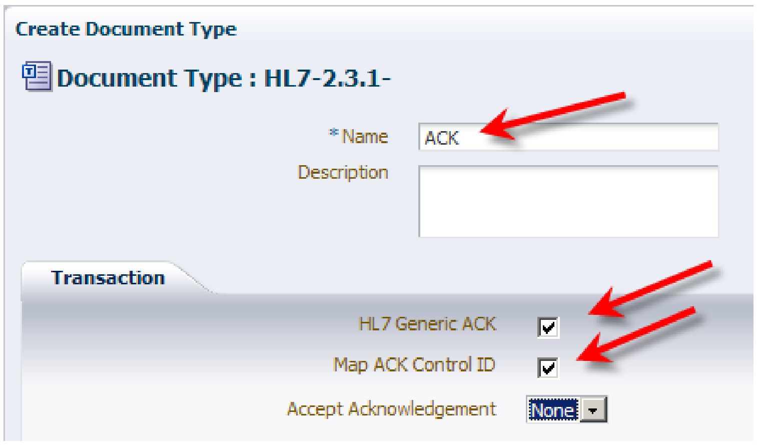
“Introduce” the ACK document to SOA Suite for healthcare integration

- ☐ Switch to the Healthcare integration Console

- ☐ Expand the “Designer”→”Configuration”→”Document Protocol”→”HL7” hierarchy, right-click the “2.3.1” node and choose the “Create” option



- ☐ Name the document type “ACK”, check the “HL7 Generic ACK” checkbox, check the “Map ACK Control ID” checkbox and click “OK” to complete the dialogue



Create Document Type

Document Type : HL7-2.3.1-

* Name	ACK
Description	

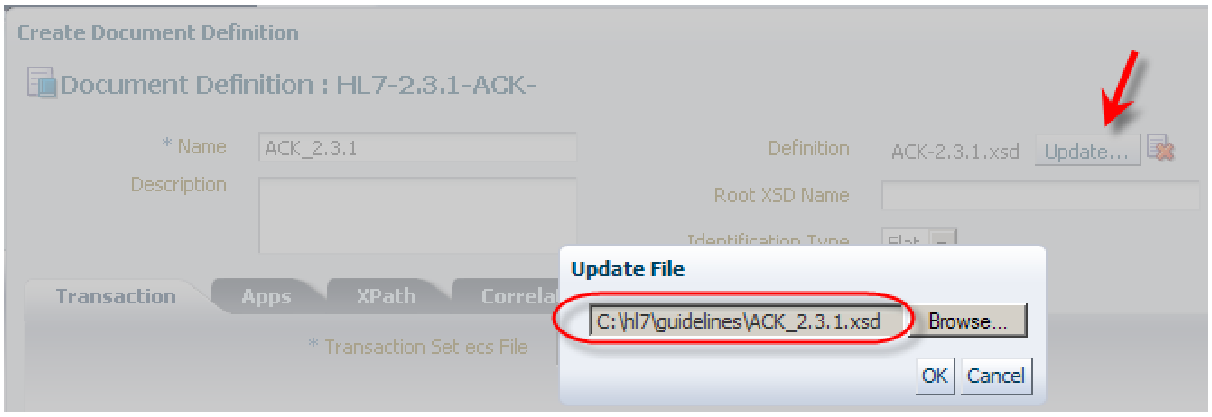
Transaction

HL7 Generic ACK	✓
Map ACK Control ID	✓
Accept Acknowledgement	None

- ☐ Expand the “2.3.1.” node, right click the “ACK” Node and choose “Create”

- ☐ Name the new document definition “ACK_2.3.1”

- ☐ Click “Update” button alongside the “Definition” label, locate the new ACK_2.3.1.xsd file and choose it



Create Document Definition

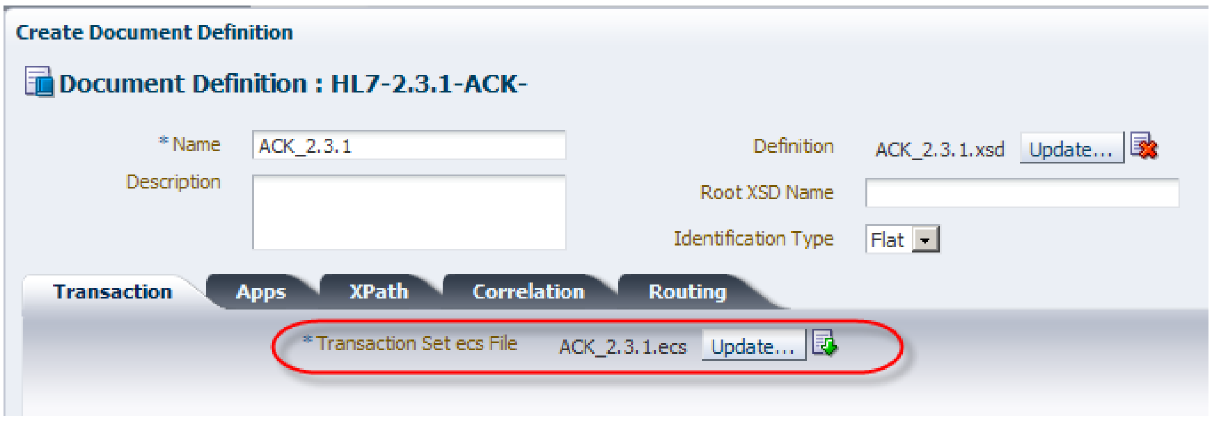
Document Definition : HL7-2.3.1-ACK-

* Name	ACK_2.3.1	Definition	ACK-2.3.1.xsd
Description		Root XSD Name	
		Identification Type	Flat

Update File

C:\hl7\guidelines\ACK_2.3.1.xsd

☐ Click the "Browse" button alongside the "Transaction Set ecs File", locate the "ACK_2.3.1.ecs" file, choose the file and click "OK" to close the dialogue box


Create Document Definition

Document Definition : HL7-2.3.1-ACK-

* Name	ACK_2.3.1	Definition	ACK_2.3.1.xsd
Description		Root XSD Name	
		Identification Type	Flat

* Transaction Set ecs File ACK_2.3.1.ecs

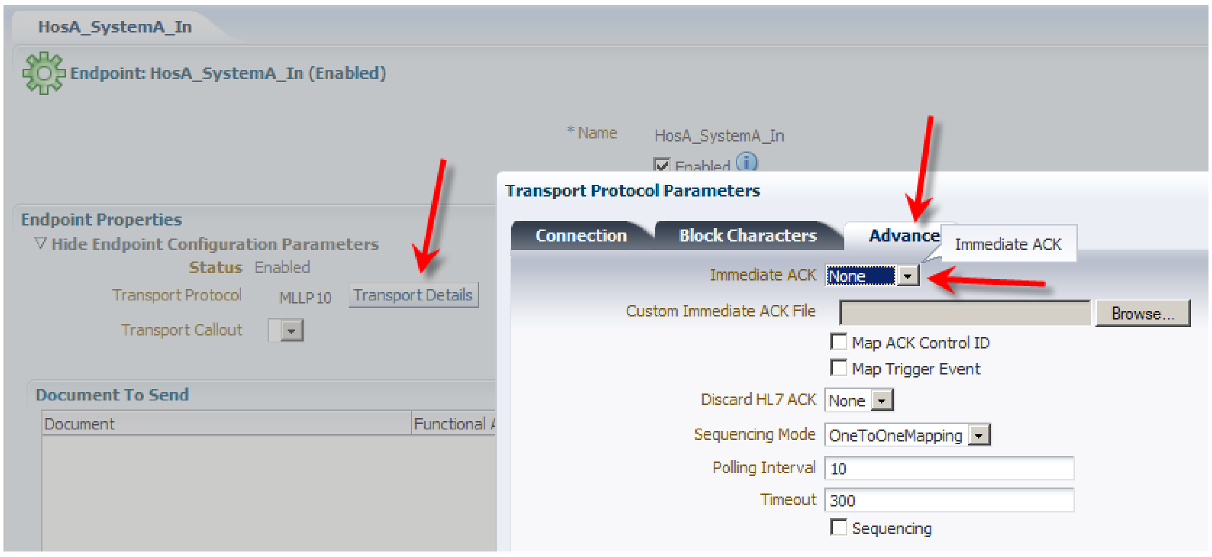
☐ Close the document-related TABs in the right hand side of the Healthcare Integration Console to reduce resource consumption

Configure Translation and Functional Acknowledgement

The ACK document is now available so we can proceed to re-configure the 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"


HosA_SystemA_In

Endpoint: HosA_SystemA_In (Enabled)

* Name HosA_SystemA_In

Enabled

Endpoint Properties

▽ Hide Endpoint Configuration Parameters

Status	Enabled
Transport Protocol	MLLP10
Transport Callout	

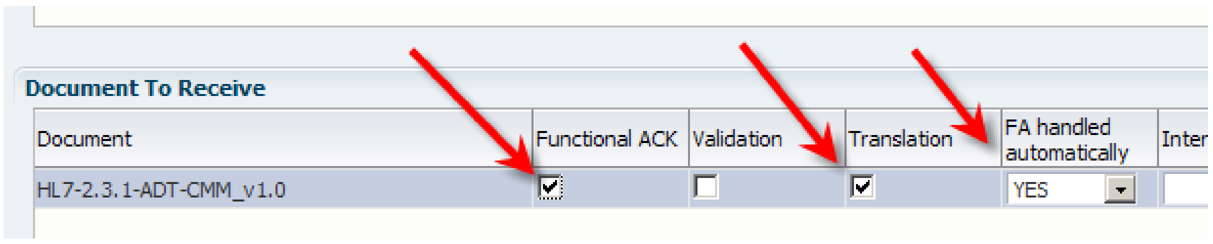
Document To Send

Document	Functional A

Transport Protocol Parameters

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

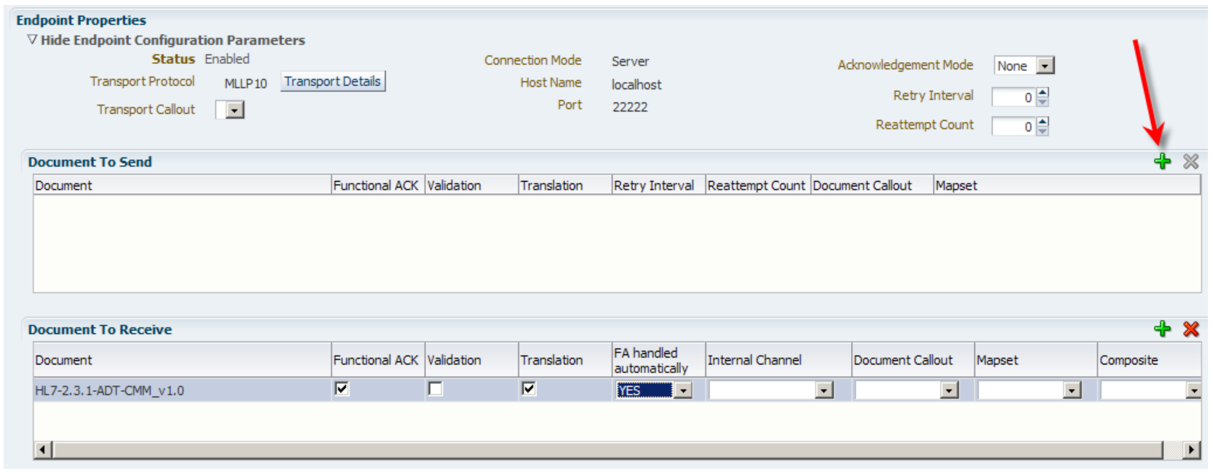
☐ In the "Document To Receive" section of the "HosA_SystemA_In" Tab check the "Functional ACK" checkbox, check the "Translation" checkbox, choose "YES" for the "FA handled automatically" dropdown



Document To Receive

Document	Functional ACK	Validation	Translation	FA handled automatically	Inter
HL7-2.3.1-ADT-CMM_v1.0	✓	☐	✓	YES	

☐ Click the and click "Add" button in the "Document To Send" section to add the ACLK document as the document which this endpoint will send



Endpoint Properties
▽ Hide Endpoint Configuration Parameters

Status	Enabled	Connection Mode	Server	Acknowledgement Mode	None
Transport Protocol	MLLP10 Transport Details	Host Name	localhost	Retry Interval	0
Transport Callout		Port	22222	Reattempt Count	0

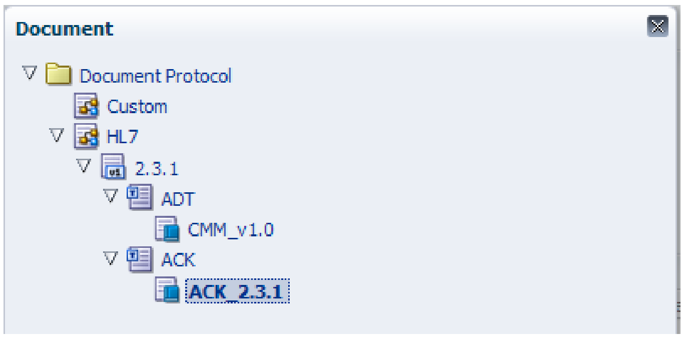
Document To Send

Document	Functional ACK	Validation	Translation	Retry Interval	Reattempt Count	Document Callout	Mapset

Document To Receive

Document	Functional ACK	Validation	Translation	FA handled automatically	Internal Channel	Document Callout	Mapset	Composite
HL7-2.3.1-ADT-CMM_v1.0	✓	☐	✓	YES				

☐ Choose the "ACK_2.3.1" document, click "OK" to dismiss the dialogue box and click "Apply" – this re-configures the endpoint to perform delimited to XML translation and return functional acknowledgements once the message is parsed –applying these changes alters the behavior of the HL7 receiver

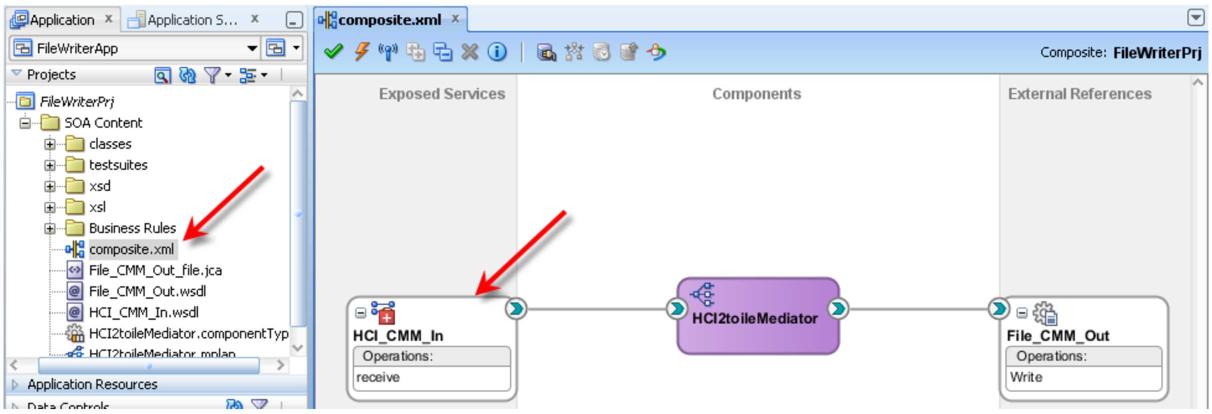


Document

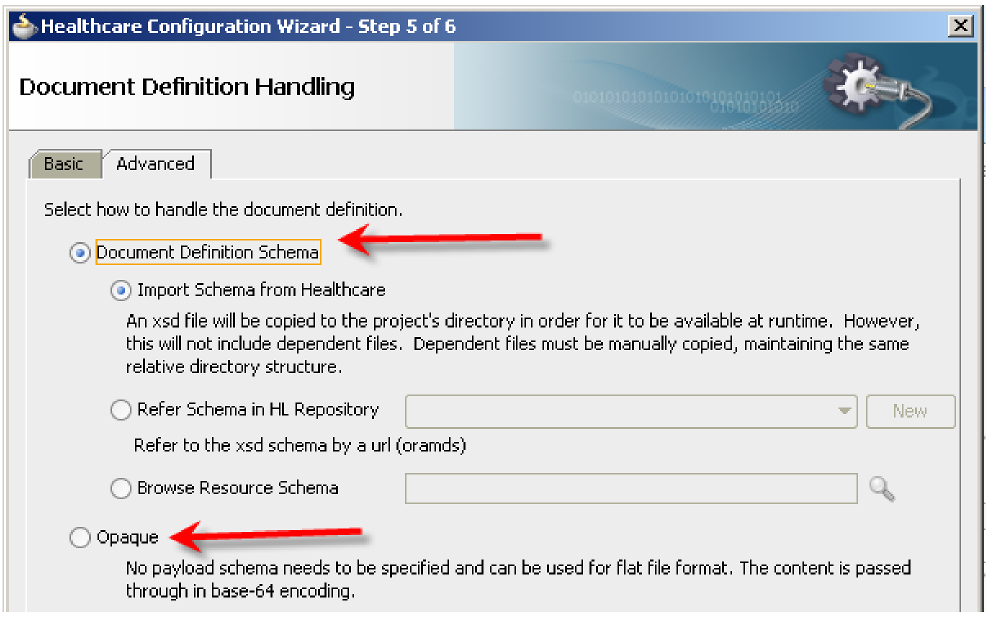
- ▽ Document Protocol
  - Custom
  - ▽ HL7
    - ▽ 2.3.1
      - ▽ ADT
        - CMM_v1.0
      - ▽ ACK
        - ACK_2.3.1

Re-configure and re-deploy the FileWriter project

☐ Start JDeveloper Studio, if not already running, locate and double-click the composite.xml to open the composite graphical editor, the double-click the HCI_CMM_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”


Healthcare Configuration Wizard - Step 5 of 6

Document Definition Handling

Basic | Advanced

Select how to handle the document definition.

[✓ Document Definition Schema]

[✓ Import Schema from Healthcare]

An xsd file will be copied to the project's directory in order for it to be available at runtime. However, this will not include dependent files. Dependent files must be manually copied, maintaining the same relative directory structure.

Refer Schema in HL Repository | New

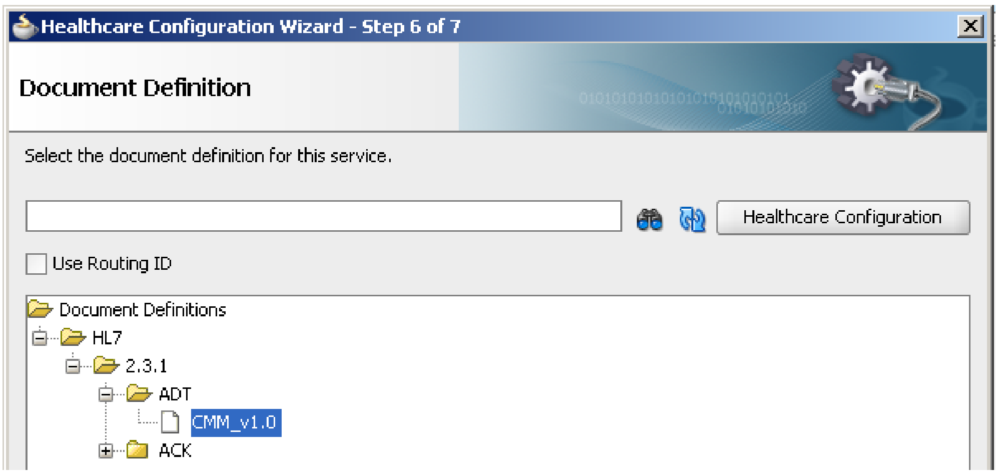
Refer to the xsd schema by a url (oramds)

Browse Resource Schema

Opaque

No payload schema needs to be specified and can be used for flat file format. The content is passed through in base-64 encoding.

☐ Select the “CMM_v1.0” document from the “”, then click “Next” and “Finish”


Healthcare Configuration Wizard - Step 6 of 7

Document Definition

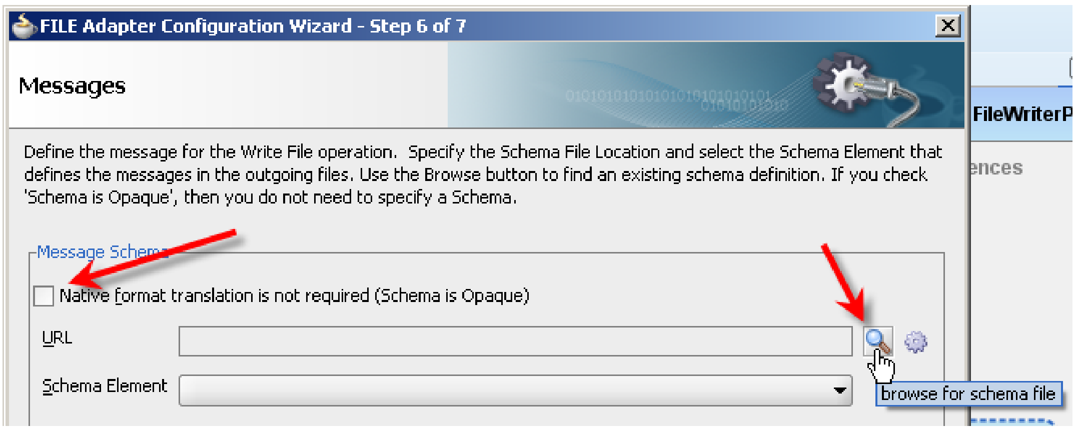
Select the document definition for this service.

Healthcare Configuration

Use Routing ID

Document Definitions
- HL7
  - 2.3.1
    - ADT
      - CMM_v1.0
    - ACK

☐ Double-click the “File_CMM_Out” adapter to start its configuration wizard, click “Next” until you see the “Message” dialogue panel, uncheck the “Native format translation is not required …” checkbox, then click the “Browse for schema file” button


FILE Adapter Configuration Wizard - Step 6 of 7

Messages

Define the message for the Write File operation. Specify the Schema File Location and select the Schema Element that defines the messages in the outgoing files. Use the Browse button to find an existing schema definition. If you check 'Schema is Opaque', then you do not need to specify a Schema.

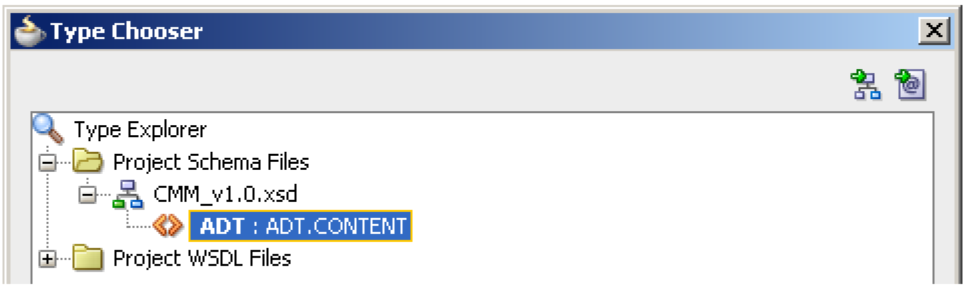
Message Schema

☐ Native format translation is not required (Schema is Opaque)

URL

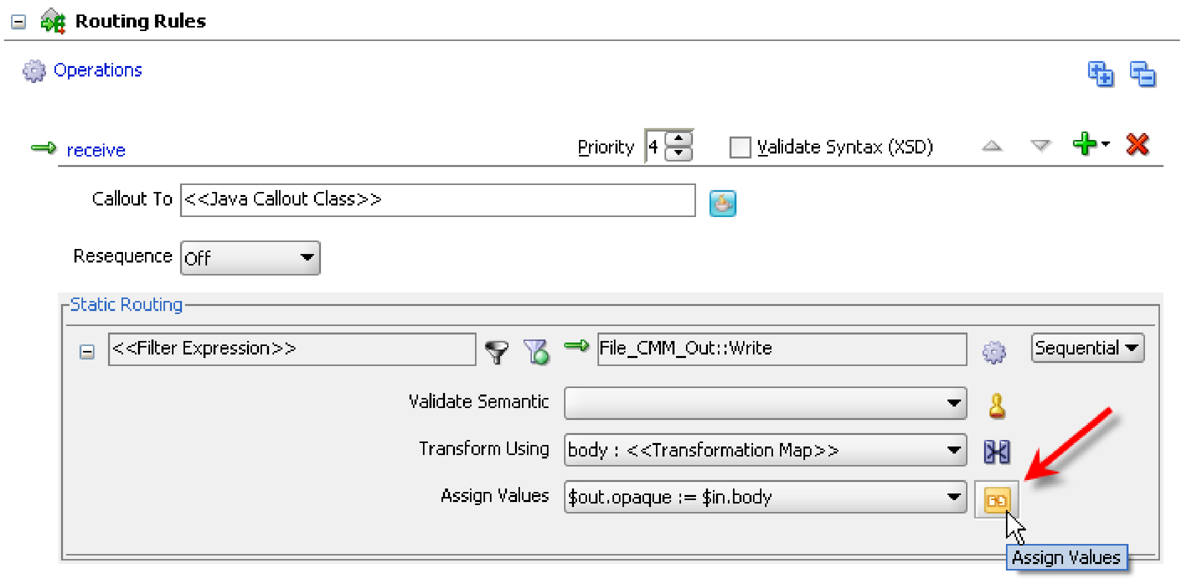
Schema Element

☐ Expand the "Project Schema Files"→"CMM_v1.0" hierarch, select the "ADT" node and click "OK" to complete and dismiss the dialogue



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

☐ Double-click the "HCI2FileMediator" mediator to open its configuration panel, then click the "Assign Values" button


Routing Rules

Operations

receive — Priority 4 — ☐ Validate Syntax (XSD)

Callout To: <<Java Callout Class>>

Resequence: Off

Static Routing

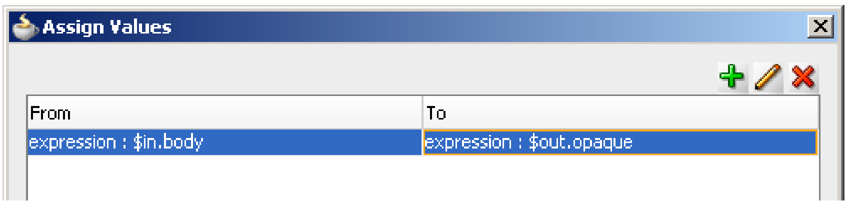
<<Filter Expression>> — File_CMM_Out::Write — Sequential

Validate Semantic:

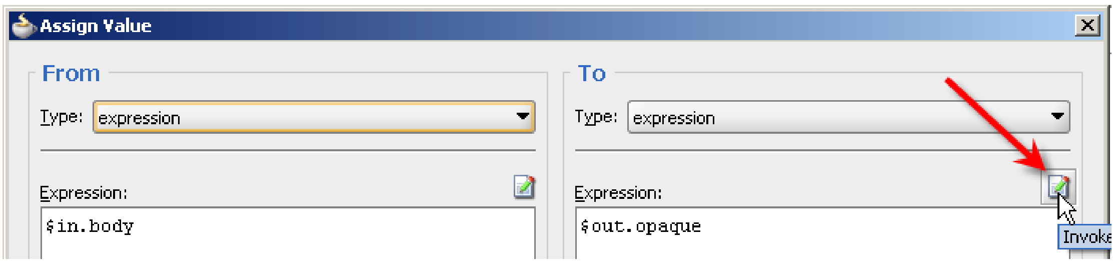
Transform Using: body : <<Transformation Map>>

Assign Values: $out.opaque := $in.body

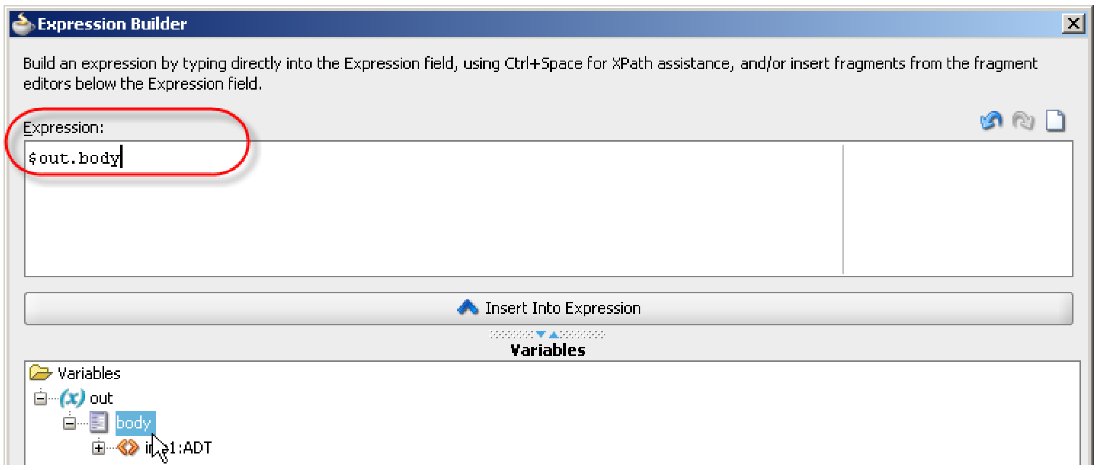
☐ Double-click the only line in the "Assign Values" dialogue to open the "Assign Value" dialogue



☐ Click the “Invoke Expression Builder” button on the “To” side

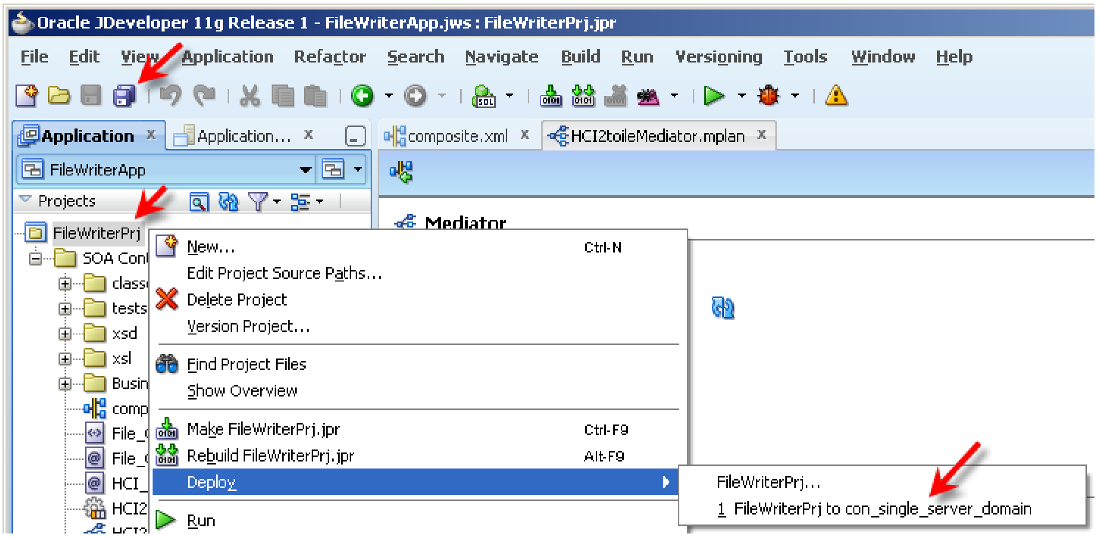


☐ Select the “$out.opaque” text in the “Expression” box and delete it, expand the “out” tree in the “Variables” pane, double-click the “body” node to add the expression “$out.body” to the “Expression” box replacing the “$out.opaque” expression, then clikc “OK’, “OK” and “OK” to complete the process



Build an expression by typing directly into the Expression field, using Ctrl+Space for XPath assistance, and/or insert fragments from the fragment editors below the Expression field.

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


Oracle JDeveloper 11g Release 1 - FileWriterApp.jws : FileWriterPrj.jpr
File Edit View Application Refactor Search Navigate Build Run Versioning Tools Window Help
Application
Application...
composite.xml
HCI2toileMediator.mplan
FileWriterApp
Projects
FileWriterPrj
SOA Con
class
tests
xsd
xsl
Busin
comp
File_
File_
HCI_
HCI2
Mediator
New... Ctrl-N
Edit Project Source Paths...
Delete Project
Version Project...
Find Project Files
Show Overview
Make FileWriterPrj.jpr Ctrl-F9
Rebuild FileWriterPrj.jpr Alt-F9
Deploy
Run
FileWriterPrj...
1 FileWriterPrj to con_single_server_domain

☐ Close JDeveloper

Process ADT messages

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 output in our configured output directory – for me c:\hl7\received - 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, and acknowledging them and writing them to files in the file system.

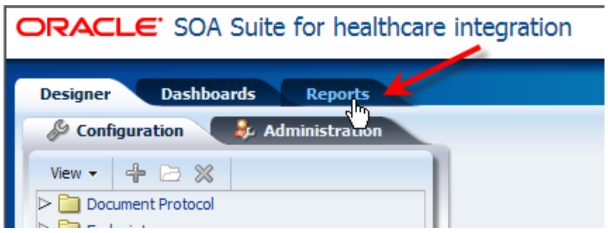
Explore Message Tracking

Let's explore message tracking.

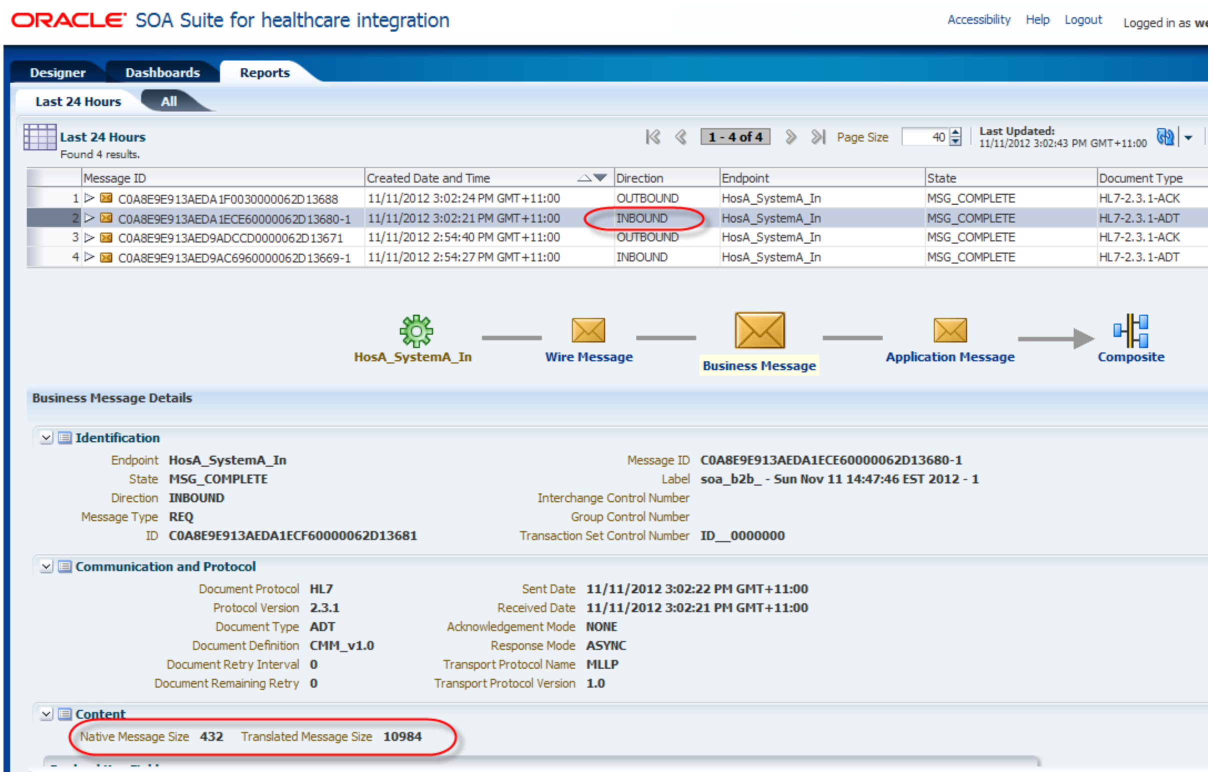
☐ Start the Healthcare Integration Console – http://localhost:7001/healthcare

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

☐ Click the "Reports" Tab



☐ Select the second-from-last message with the "Direction" of "INBOUND" in the list, 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



Last 24 Hours
Found 4 results.

Last Updated: 11/11/2012 3:02:43 PM GMT+11:00

	Message ID	Created Date and Time	Direction	Endpoint	State	Document Type
1	C0A8E9E913AEDA1F0030000062D13688	11/11/2012 3:02:24 PM GMT+11:00	OUTBOUND	HosA_SystemA_In	MSG_COMPLETE	HL7-2.3.1-ACK
2	C0A8E9E913AEDA1ECE60000062D13680-1	11/11/2012 3:02:21 PM GMT+11:00	INBOUND	HosA_SystemA_In	MSG_COMPLETE	HL7-2.3.1-ADT
3	C0A8E9E913AED9ADCCD0000062D13671	11/11/2012 2:54:40 PM GMT+11:00	OUTBOUND	HosA_SystemA_In	MSG_COMPLETE	HL7-2.3.1-ACK
4	C0A8E9E913AED9AC6960000062D13669-1	11/11/2012 2:54:27 PM GMT+11:00	INBOUND	HosA_SystemA_In	MSG_COMPLETE	HL7-2.3.1-ADT

Business Message Details

Identification

Endpoint	HosA_SystemA_In	Message ID	C0A8E9E913AEDA1ECE60000062D13680-1
State	MSG_COMPLETE	Label	soa_b2b_ - Sun Nov 11 14:47:46 EST 2012 - 1
Direction	INBOUND	Interchange Control Number	
Message Type	REQ	Group Control Number	
ID	C0A8E9E913AEDA1ECF60000062D13681	Transaction Set Control Number	ID__0000000

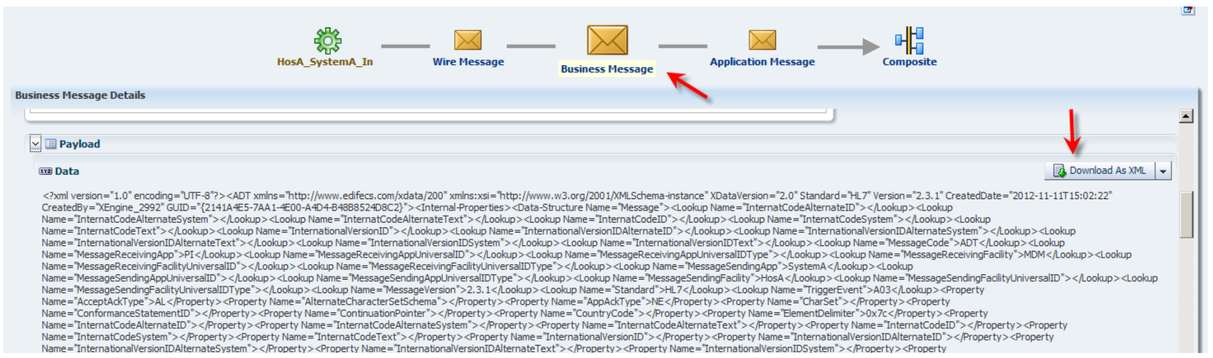
Communication and Protocol

Document Protocol	HL7	Sent Date	11/11/2012 3:02:22 PM GMT+11:00
Protocol Version	2.3.1	Received Date	11/11/2012 3:02:21 PM GMT+11:00
Document Type	ADT	Acknowledgement Mode	NONE
Document Definition	CMM_v1.0	Response Mode	ASYNC
Document Retry Interval	0	Transport Protocol Name	MLLP
Document Remaining Retry	0	Transport Protocol Version	1.0

Content

Native Message Size 432 Translated Message Size 10984

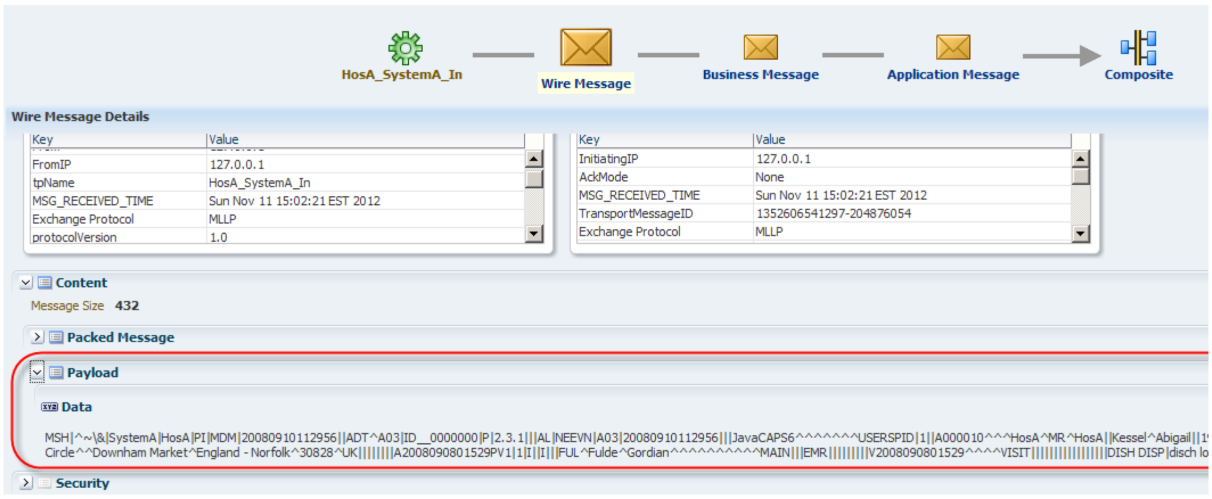
☐ Scroll down the "Business Message Details" pane, expand the "Payload" node and inspect the HL7 message payload, noting the "Download as XML" button, which allows the payload to be "externalized", i.e. saved into a file in the file system – the "Business Message" is the translated message in the XML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



Business Message Details

<?xml version="1.0" encoding="UTF-8"?><ADT xmlns="http://www.edifecs.com/xdata/200" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" XDataVersion="2.0" Standard="HL7" Version="2.3.1" CreatedDate="2012-11-11T15:02:22"
CreatedBy="XEngine_2992" GUID="{2141A4E5-7AA1-4E00-A4D4-B48B8524D8C2}"><Internal-Properties><Data-Structure Name="Message"><Lookup Name="InternatCodeAlternateID"></Lookup><Lookup
Name="InternatCodeAlternateSystem"></Lookup><Lookup Name="InternatCodeAlternateText"></Lookup><Lookup Name="InternatCodeID"></Lookup><Lookup Name="InternatCodeSystem"></Lookup><Lookup
Name="InternatCodeText"></Lookup><Lookup Name="InternationalVersionID"></Lookup><Lookup Name="InternationalVersionIDAlternateID"></Lookup><Lookup Name="InternationalVersionIDAlternateSystem"></Lookup><Lookup
Name="InternationalVersionIDAlternateText"></Lookup><Lookup Name="InternationalVersionIDSystem"></Lookup><Lookup Name="InternationalVersionIDText"></Lookup><Lookup Name="MessageCode">ADT</Lookup><Lookup
Name="MessageReceivingApp">PI</Lookup><Lookup Name="MessageReceivingAppUniversalID"></Lookup><Lookup Name="MessageReceivingAppUniversalIDType"></Lookup><Lookup Name="MessageReceivingFacility">MDM</Lookup><Lookup
Name="MessageReceivingFacilityUniversalID"></Lookup><Lookup Name="MessageReceivingFacilityUniversalIDType"></Lookup><Lookup Name="MessageSendingApp">SystemA</Lookup><Lookup
Name="MessageSendingAppUniversalID"></Lookup><Lookup Name="MessageSendingAppUniversalIDType"></Lookup><Lookup Name="MessageSendingFacility">HosA</Lookup><Lookup Name="MessageSendingFacilityUniversalID"></Lookup><Lookup
Name="MessageSendingFacilityUniversalIDType"></Lookup><Lookup Name="MessageVersion">2.3.1</Lookup><Lookup Name="Standard">HL7</Lookup><Lookup Name="TriggerEvent">A03</Lookup><Property
Name="AcceptAckType">AL</Property><Property Name="AlternateCharacterSetSchema"></Property><Property Name="AppAckType">NE</Property><Property Name="CharSet"></Property><Property
Name="ConformanceStatementID"></Property><Property Name="ContinuationPointer"></Property><Property Name="CountryCode"></Property><Property Name="ElementDelimiter">0x7c</Property><Property
Name="InternatCodeAlternateID"></Property><Property Name="InternatCodeAlternateSystem"></Property><Property Name="InternatCodeAlternateText"></Property><Property Name="InternatCodeID"></Property><Property
Name="InternatCodeSystem"></Property><Property Name="InternatCodeText"></Property><Property Name="InternationalVersionID"></Property><Property Name="InternationalVersionIDAlternateID"></Property><Property
Name="InternationalVersionIDAlternateSystem"></Property><Property Name="InternationalVersionIDAlternateText"></Property><Property Name="InternationalVersionIDSystem"></Property><Property

☐ 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



Wire Message Details

Key	Value
FromIP	127.0.0.1
tpName	HosA_SystemA_In
MSG_RECEIVED_TIME	Sun Nov 11 15:02:21 EST 2012
Exchange Protocol	MLLP
protocolVersion	1.0

Key	Value
InitiatingIP	127.0.0.1
AckMode	None
MSG_RECEIVED_TIME	Sun Nov 11 15:02:21 EST 2012
TransportMessageID	1352606541297-204876054
Exchange Protocol	MLLP

Content

Message Size 432

Packed Message

Payload

Data

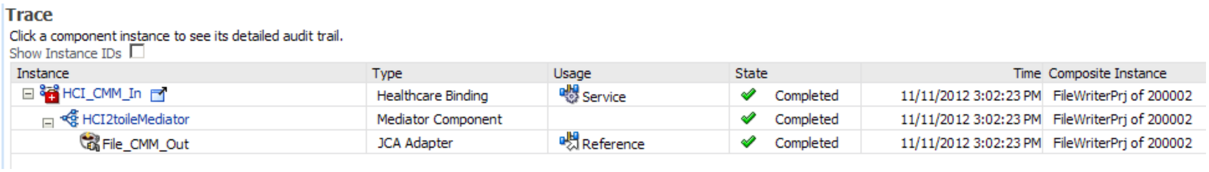
```
MSH|^~\&|SystemA|HosA|PI|MDM|20080910112956||ADT^A03|ID__0000000|P|2.3.1|||AL|NEEVN|A03|20080910112956|||JavaCAPS6^^^^^^^USERSPID|1||A000010^^^HosA^MR^HosA||Kessel^Abigail||1
Circle^^Downham Market^England - Norfolk^30828^UK|||||||A2008090801529PV1|1|I||I|||FUL^Fulde^Gordian^^^^^^^^^^MAIN|||EMR||||||||V2008090801529^^^^VISIT|||||||||||||||DISH DISP|disch lo
```

Security

☐ 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



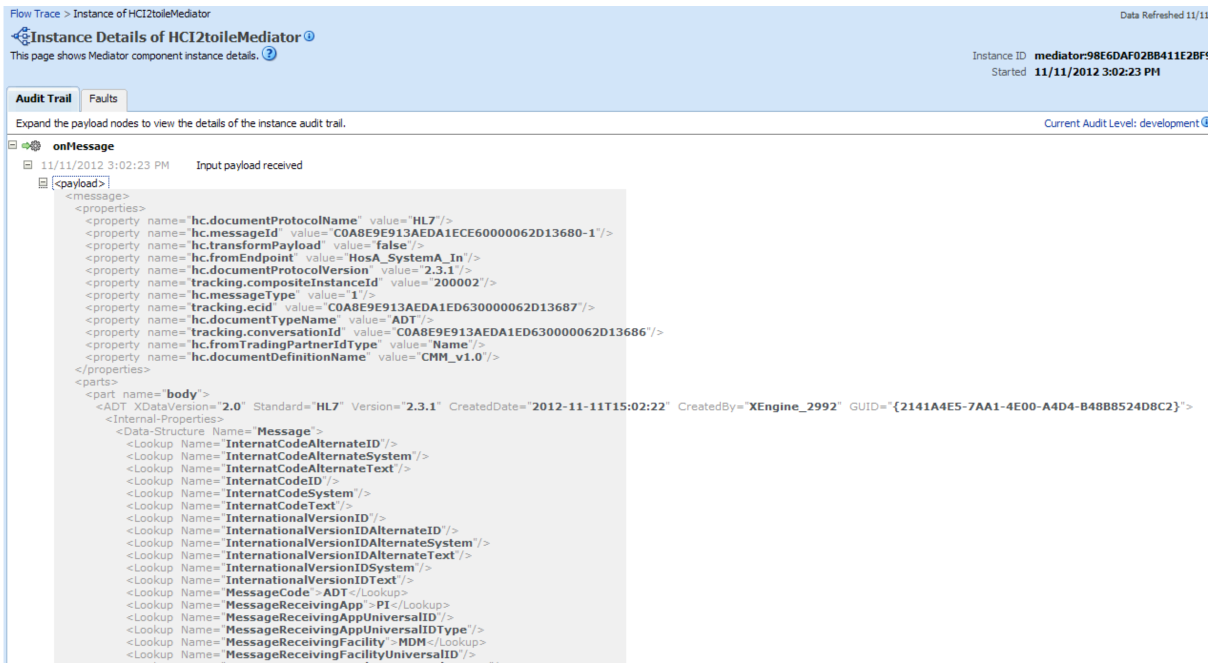
Trace

Click a component instance to see its detailed audit trail.

Show Instance IDs ☐

Instance	Type	Usage	State	Time	Composite Instance
⊟ HCI_CMM_In	Healthcare Binding	Service	✓ Completed	11/11/2012 3:02:23 PM	FileWriterPrj of 200002
⊟ HCI2toileMediator	Mediator Component		✓ Completed	11/11/2012 3:02:23 PM	FileWriterPrj of 200002
File_CMM_Out	JCA Adapter	Reference	✓ Completed	11/11/2012 3:02:23 PM	FileWriterPrj of 200002

☐ Click the "HCI2FileMediator" 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


Flow Trace > Instance of HCI2toileMediator

Data Refreshed 11/11

Instance Details of HCI2toileMediator

This page shows Mediator component instance details.

Instance ID mediator:98E6DAF02BB411E2BF

Started 11/11/2012 3:02:23 PM

Audit Trail | Faults

Expand the payload nodes to view the details of the instance audit trail.

Current Audit Level: development

```
onMessage
  11/11/2012 3:02:23 PM     Input payload received
    <payload>
      <message>
        <properties>
          <property name="hc.documentProtocolName" value="HL7"/>
          <property name="hc.messageId" value="C0A8E9E913AEDA1ECE60000062D13680-1"/>
          <property name="hc.transformPayload" value="false"/>
          <property name="hc.fromEndpoint" value="HosA_SystemA_In"/>
          <property name="hc.documentProtocolVersion" value="2.3.1"/>
          <property name="tracking.compositeInstanceId" value="200002"/>
          <property name="hc.messageType" value="1"/>
          <property name="tracking.ecid" value="C0A8E9E913AEDA1ED630000062D13687"/>
          <property name="hc.documentTypeName" value="ADT"/>
          <property name="tracking.conversationId" value="C0A8E9E913AEDA1ED630000062D13686"/>
          <property name="hc.fromTradingPartnerIdType" value="Name"/>
          <property name="hc.documentDefinitionName" value="CMM_v1.0"/>
        </properties>
        <parts>
          <part name="body">
            <ADT XDataVersion="2.0" Standard="HL7" Version="2.3.1" CreatedDate="2012-11-11T15:02:22" CreatedBy="XEngine_2992" GUID="{2141A4E5-7AA1-4E00-A4D4-B48B8524D8C2}">
              <Internal-Properties>
                <Data-Structure Name="Message">
                  <Lookup Name="InternatCodeAlternateID"/>
                  <Lookup Name="InternatCodeAlternateSystem"/>
                  <Lookup Name="InternatCodeAlternateText"/>
                  <Lookup Name="InternatCodeID"/>
                  <Lookup Name="InternatCodeSystem"/>
                  <Lookup Name="InternatCodeText"/>
                  <Lookup Name="InternationalVersionID"/>
                  <Lookup Name="InternationalVersionIDAlternateID"/>
                  <Lookup Name="InternationalVersionIDAlternateSystem"/>
                  <Lookup Name="InternationalVersionIDAlternateText"/>
                  <Lookup Name="InternationalVersionIDSystem"/>
                  <Lookup Name="InternationalVersionIDText"/>
                  <Lookup Name="MessageCode">ADT</Lookup>
                  <Lookup Name="MessageReceivingApp">PI</Lookup>
                  <Lookup Name="MessageReceivingAppUniversalID"/>
                  <Lookup Name="MessageReceivingAppUniversalIDType"/>
                  <Lookup Name="MessageReceivingFacility">MDM</Lookup>
                  <Lookup Name="MessageReceivingFacilityUniversalID"/>
```

- ☐ Explicitly open the Enterprise Manager Console - http://localhost:7001/em - and click
- ☐ Close the Enterprise Manager Console windows
- ☐ Switch back to Healthcare integration Console
- ☐ Select one of the "OUTBOUND" ACK Messages and inspect the Wire and Business message forms – note that the "Application Message" icon is inaccessible because the ACK has been generated by the Healthcare Integration infrastructure rather than being provided by the SOA Composite

HL7 v2 Inbound to file with explicit name

In the previous section we received a HL7 v2 delimited message and wrote it as-is to a file in the file system with translation to XML. The name of the file was set in the File Adapter to a name which was generated to be unique using a timestamp pattern. In this section we will re-configure the SOA Composite to cause the file name to be constructed using message content, messaging environment and endpoint properties. This has nothing to do with HL7 recept to XML transformation but illustrates that the message content and messaging properties can be used inside the SOA Composite for whatever processing might be required.

We will construct the file name using the following elements, with literal '_' between elements, concatenating elements to form a name of the file, for example HL7_2.3.1_ADT_....xml.

Document environment properties:

hc.documentProtocolName

hc.documentProtocolVersion

hc.documentTypeName

Document Content values

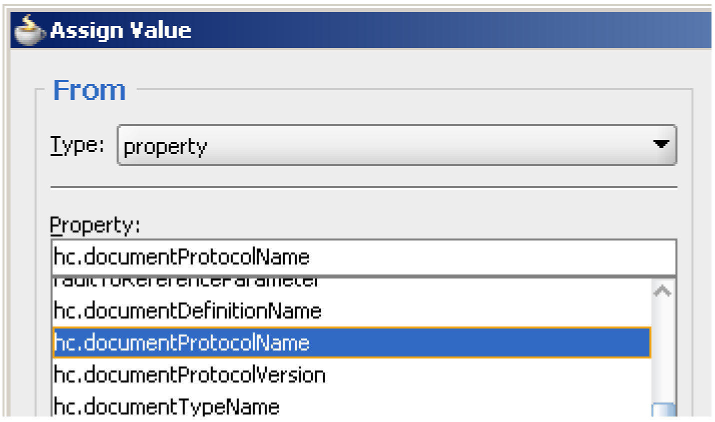
MSH.10

MSH.9

Literal string

'.xml'

Where the document content values are selectable graphically in the Expression Builder, the property expressions and literal strings must be manually entered. In the Expression Builder, a reference to a property associated with the incoming document will have the form of "$in.property.*propertyName*", where *propertyName* will be the name shown in the list of properties in the "Assign Value" dialogue, for example hc.documentProtocolName, as shown in the illustration below.



hc.documentProtocolName

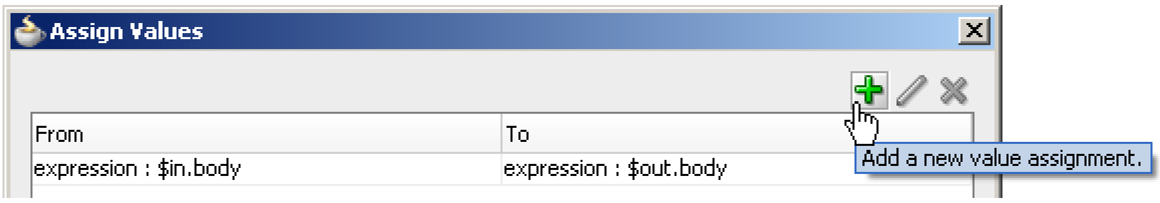
hc.documentDefinitionName
hc.documentProtocolName
hc.documentProtocolVersion
hc.documentTypeName

The complete property reference will be "$in.property.hc.documentProtocolName".

☐ Open JDeveloper Studio, if not already running, double-click the composite.xml in the FileWritrerPrj hierarch to open the composite editor and double-click the HCI2FileMediator to open mediator editor

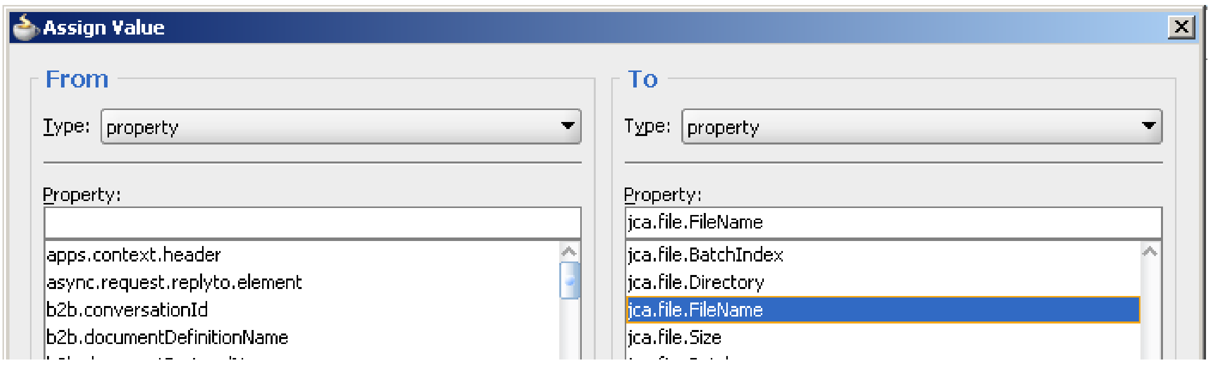
☐ Click the "Assign Values" button to open the "Assign Values" dialogue

☐ Click the "Add new value assignment" button to open the "Assign Value" dialogue



From	To
expression : $in.body	expression : $out.body

☐ In the "To" side select the "jca.file.FileName" property – we want to explicitly set the name of the file the file adapter will use, overriding its static configuration

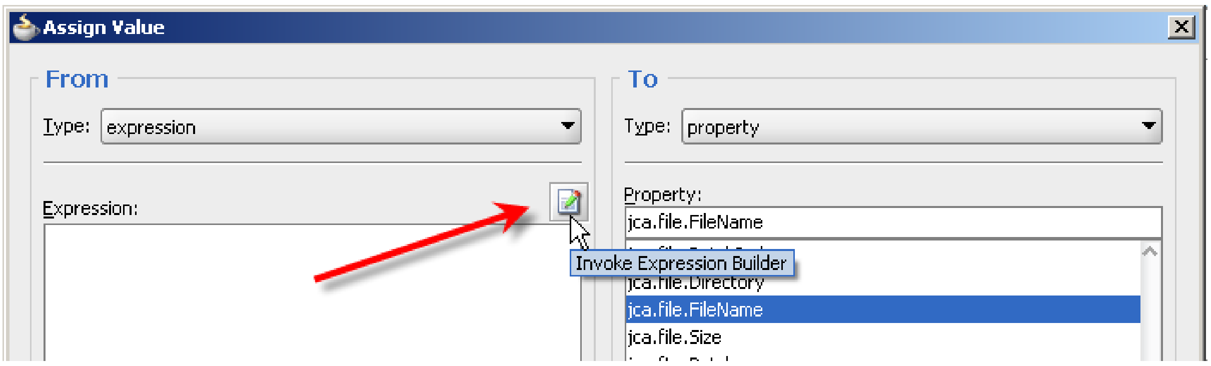


apps.context.header
async.request.replyto.element
b2b.conversationId
b2b.documentDefinitionName

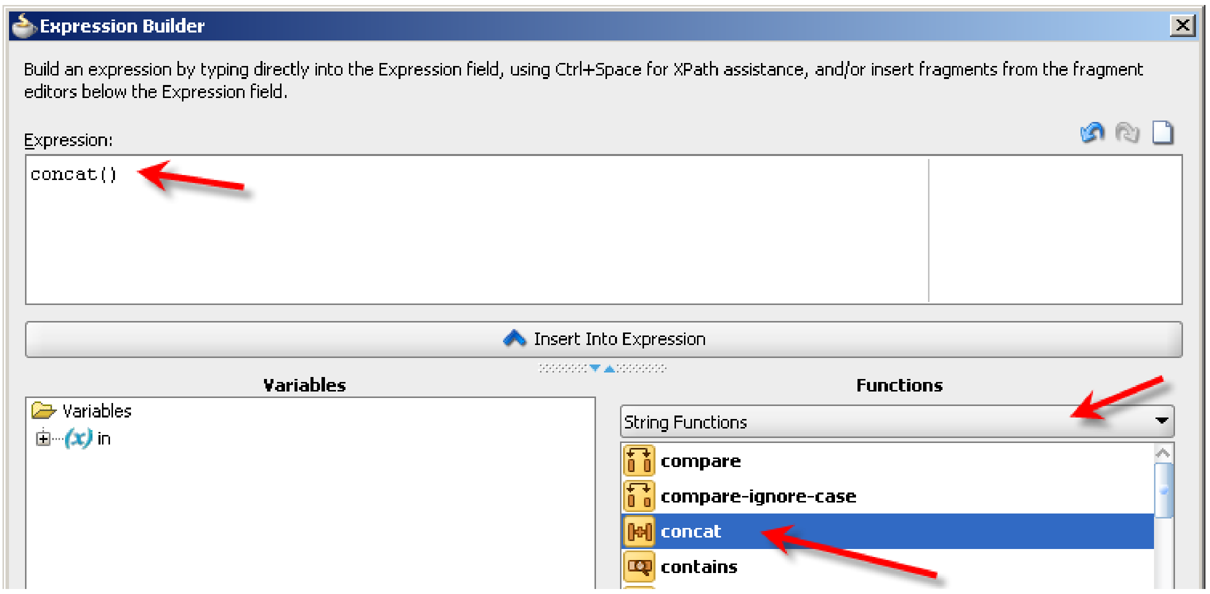
jca.file.FileName

jca.file.BatchIndex
jca.file.Directory
jca.file.FileName
jca.file.Size

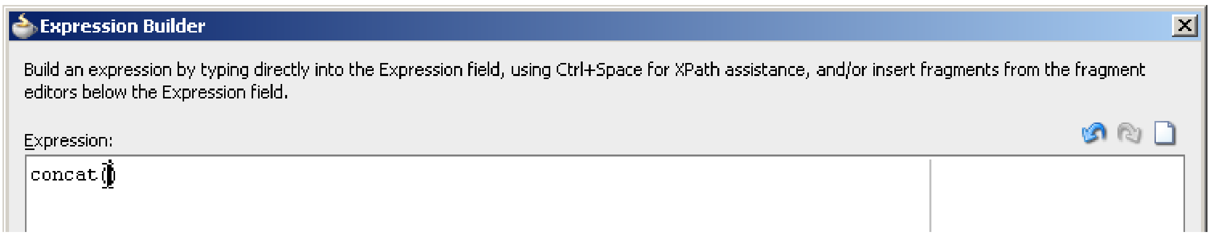
☐ In the "From" side choose the "expression" in the "Type" dropdown, then clock the "Involve Expression Builder" button



☐ In the "Functions" dropdown scroll to "String Functions" and double-click the "concat" function to add it to the Expression box


Expression Builder
Build an expression by typing directly into the Expression field, using Ctrl+Space for XPath assistance, and/or insert fragments from the fragment editors below the Expression field.
Expression:
concat()
Insert Into Expression
Variables
Variables
in
Functions
String Functions
compare
compare-ignore-case
concat
contains

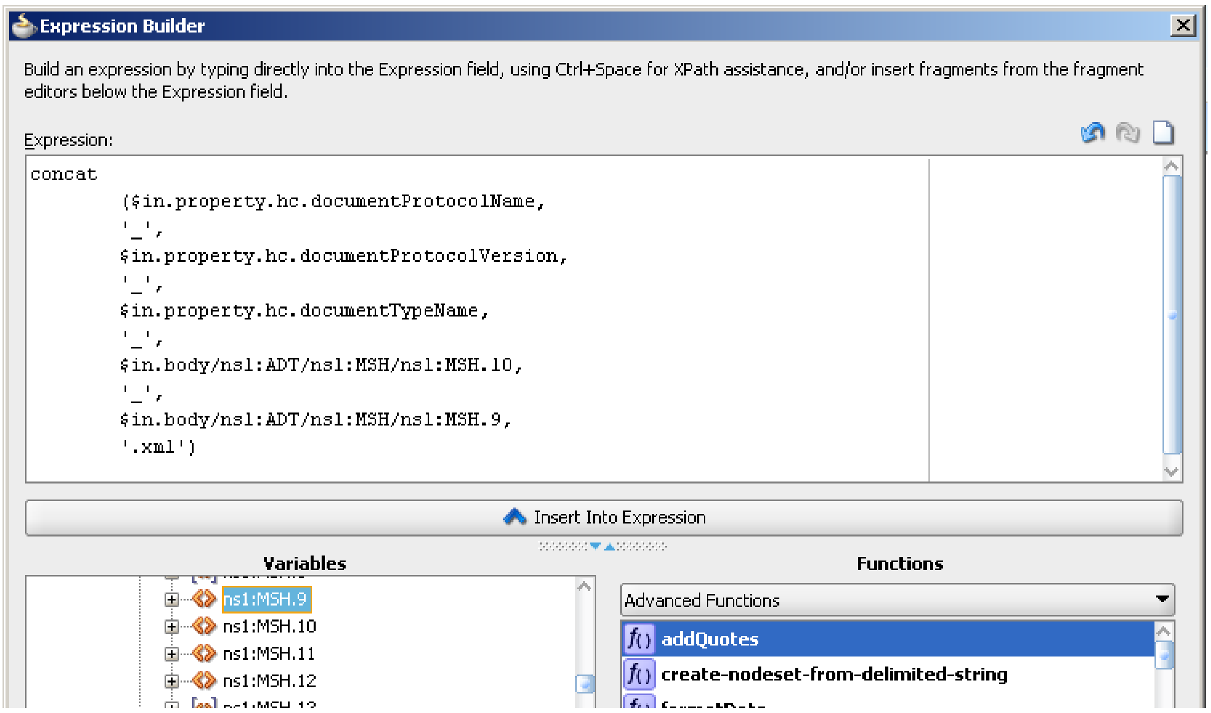
☐ Place the cursor between the parenthesis in the "Expression" box – we will be adding expression components inside the "concat" function


Expression Builder
Build an expression by typing directly into the Expression field, using Ctrl+Space for XPath assistance, and/or insert fragments from the fragment editors below the Expression field.
Expression:
concat()

☐ Select or type expression elements to construct the file name value like:

```
concat
    ($in.property.hc.documentProtocolName,
    '_',
    $in.property.hc.documentProtocolVersion,
    '_',
    $in.property.hc.documentTypeName,
```

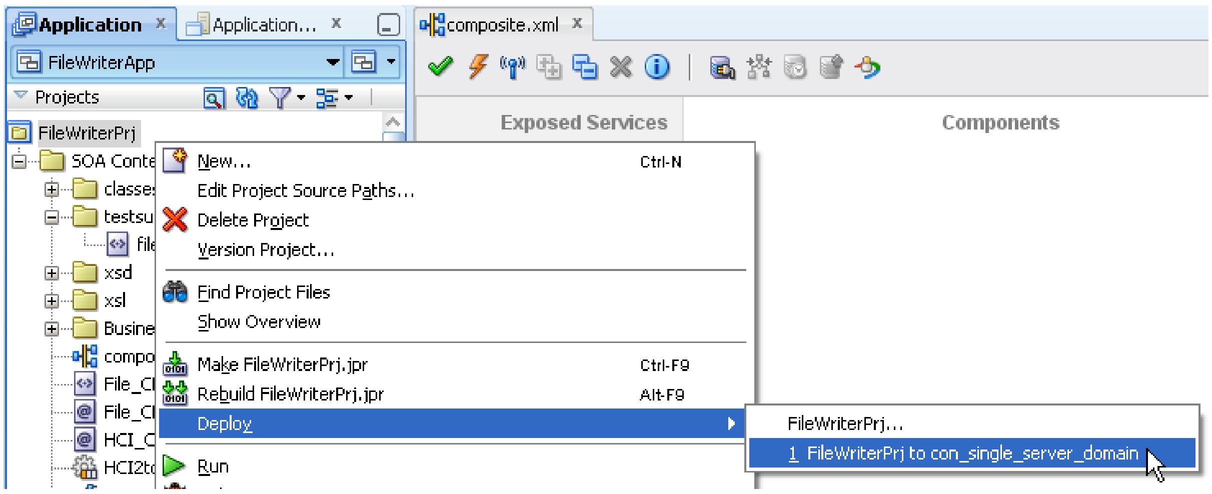
'_',
$in.body/ns1:ADT/ns1:MSH/ns1:MSH.10,
'_',
$in.body/ns1:ADT/ns1:MSH/ns1:MSH.9,
'.xml')


Expression Builder
Build an expression by typing directly into the Expression field, using Ctrl+Space for XPath assistance, and/or insert fragments from the fragment editors below the Expression field.
Expression:
Insert Into Expression
Variables
Functions
Advanced Functions
addQuotes
create-nodeset-from-delimited-string
ns1:MSH.9
ns1:MSH.10
ns1:MSH.11
ns1:MSH.12

```
concat
        ($in.property.hc.documentProtocolName,
        '_',
        $in.property.hc.documentProtocolVersion,
        '_',
        $in.property.hc.documentTypeName,
        '_',
        $in.body/ns1:ADT/ns1:MSH/ns1:MSH.10,
        '_',
        $in.body/ns1:ADT/ns1:MSH/ns1:MSH.9,
        '.xml')
```

☐ Click "OK" to dismiss the Expression Builder, click "OK" to dismiss the "Assign Value" dialogue, click "OK" to dismiss the "Assign Values" dialogue

☐ Click "Save All" JDeveloper toolbar button and deploy the application

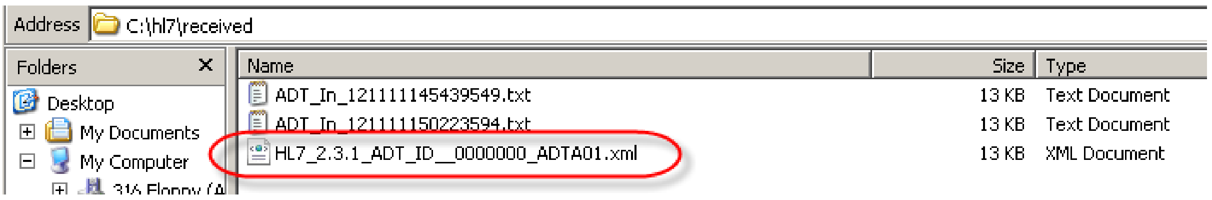


☐ Close JDeveloper

☐ 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, inspect the file name and inspect file content


Address C:\hl7\received

Name	Size	Type
ADT_In_121111145439549.txt	13 KB	Text Document
ADT_In_121111150223594.txt	13 KB	Text Document
HL7_2.3.1_ADT_ID__0000000_ADTA01.xml	13 KB	XML Document

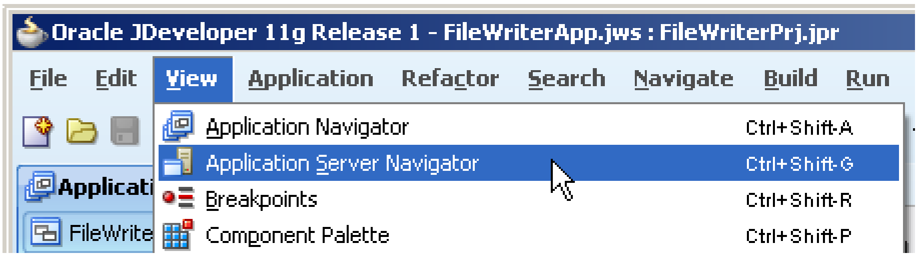
Clearly, the properties and message content can be used in the SOA Composite as might be required, perhaps affecting routing or transformation decisions.

Undeploy the FileWriter Project

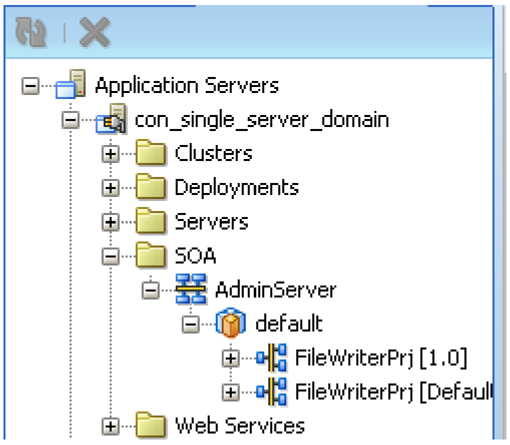
In our future projects we will re-use the HL7 receiver but rather than writing messages to files we will transform, route and send them out using outbound HL7 endpoints. We will now undeploy the File Writer project using JDeveloper.

☐ Start JDeveloper Studio if not already running

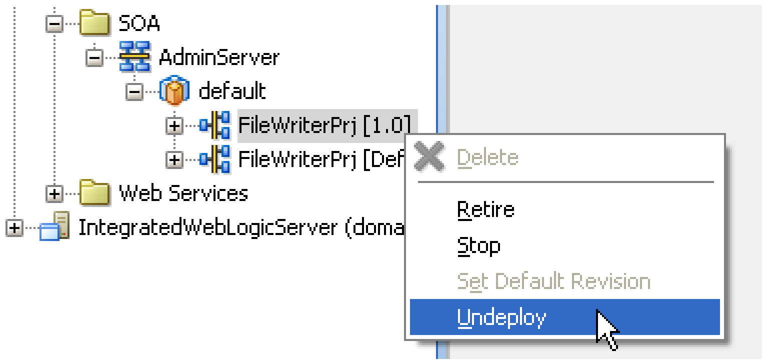
☐ Pull down the "View" menu and choose "Application Server Navigator"


Oracle JDeveloper 11g Release 1 - FileWriterApp.jws : FileWriterPrj.jpr
File
Edit
View
Application
Refactor
Search
Navigate
Build
Run
Application Navigator Ctrl+Shift-A
Application Server Navigator Ctrl+Shift-G
Breakpoints Ctrl+Shift-R
Component Palette Ctrl+Shift-P
Applicati
FileWrite

☐ Expand the "Application Server"→"*your app server connection name*"→"SOA"→"*your server name*"→"default" hierarchy



☐ Right-click the name of the project – "FileWriterPrj" and choose "Undeploy"



☐ Pull down the "Application" menu and choose "Close" to close the FileWriterApp application

We are done with the inbound HL7 v2 delimited to file example. Future examples will explore outbound adapter, 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 and configuring the SOA Suite to write each message to a file in the file system – a quintessential "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.