Oracle SOA Suite 11g

Oracle SOA Suite 11g HL7 Inbound Example

michael.czapski@oracle.com

June 2010

Table of Contents

Introduction....1
Pre-requisites....1
Prepare HL7 Data....1
Obtain and Explore the HL7 Browser....2
HL7 v2 Receiver Solution....6
Develop HL7 v2 Inbound Solution....7
Extract HL7 Message Structures....7
Configure B2B Partnership....12
Develop File Writer Solution....20
Exercise HL7 Inbound solution....36
Explore Message Tracking....40
Explore Messaging Metrics....45
Inspecting the Server Log....48
Summary....49
References....49

Introduction

As Sun Microsystems, and SeeBeyond before it, Oracle provides support for integration of systems which use HL7 v2.x messaging. Unlike Sun, and SeeBeyond before it, Oracle treats HL7 messaging as Business to Business exchanges (B2B) and uses the B2B part of the Oracle SOA Suite to accomplish the task [1].

In this article I develop and exercise a simple Oracle SOA Suite 11g B2B infrastructure-based HL7 v2 Receiver project for an ADT A01 message and use Message tracker to view messages, message states and messaging performance.

Pre-requisites

It is assumed that a Windows XP SP3 platform with the Oracle SOA Suite 11g, installed and configured as discussed in "Installing Oracle SOA Suite for HL7 Exploration", published at http://blogs.czapski.id.au/wp-content/uploads/2010/06/01_Installing_Oracle_SOA_Suite_for_HL7_exploration_v1.1.pdf, is available and will be used for the work discussed in this article.

Prepare HL7 Data

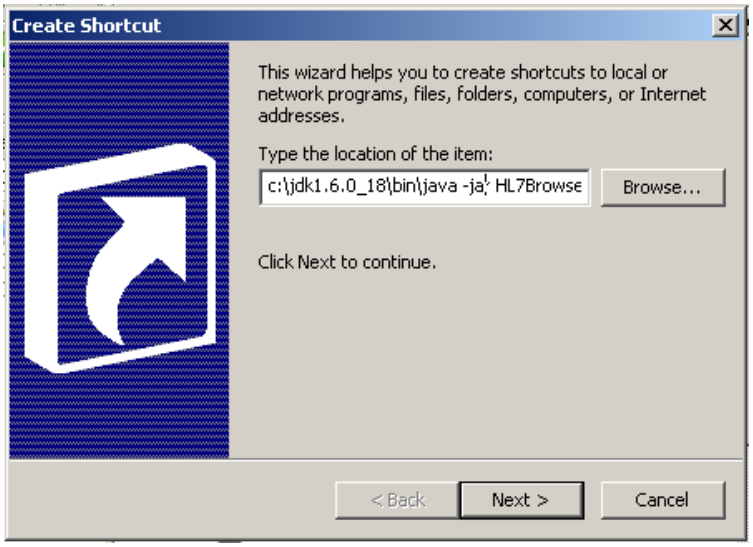
Unzip the archive, HL7_messages_sources.zip to C:\hl7\adt\data\. This archive is available from http://blogs.czapski.id.au/wp-content/uploads/2010/06/HL7_messages_sources.zip.

Obtain and Explore the HL7 Browser

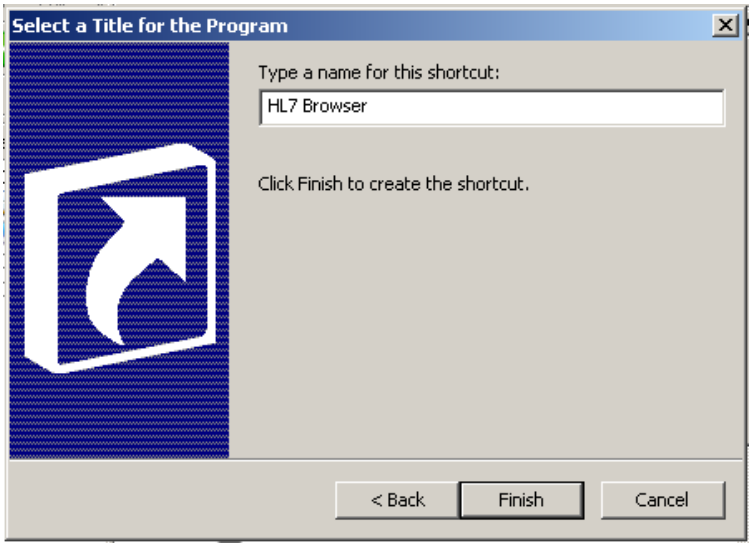
Typically a HL7 application sends or receives messages to or from another HL7 application. To keep the exercise simple we will use a 3rd party HL7 Browser to implement a sender application. This way we only need to configure the SOA Suite to receive messages.

Download and configure the free HL7 Browser tool, "HL7 Browser 1.0", from the author's page at http://mac.softpedia.com/developer/Michael-Litherland-5914.html. There are a number of other free HL7 tools by the same author at http://nule.org/wp/.

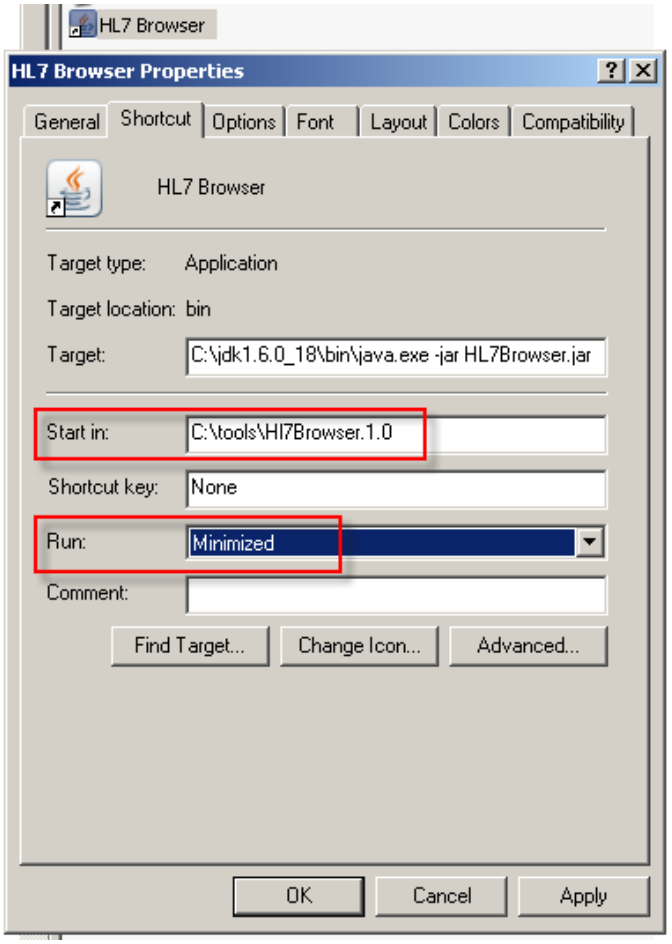
Unzip the archive, Hl7Browser.1.0.zip, to c:\tools\Hl7Browser.1.0.
Using a Windows Explorer navigate to c:\tools\Hl7Browser.1.0, right-click inside the folder that contains the HL7Browser.jar and choose New→Shortcut. Enter "c:\jdk1.6.0_18\bin\java -jar C:\tools\Hl7Browser.1.0\HL7Browser.jar" as "location of the item" and click Next. (This assumes that JDK 1.6.0_18 has been installed previously)



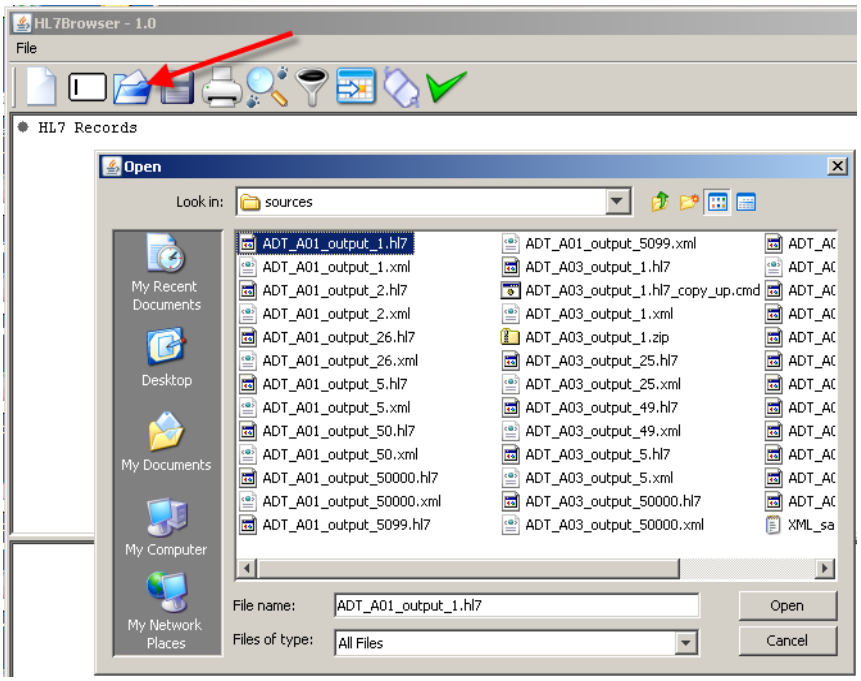
Enter "HL7 Browser" as the name of the item and click Finish.



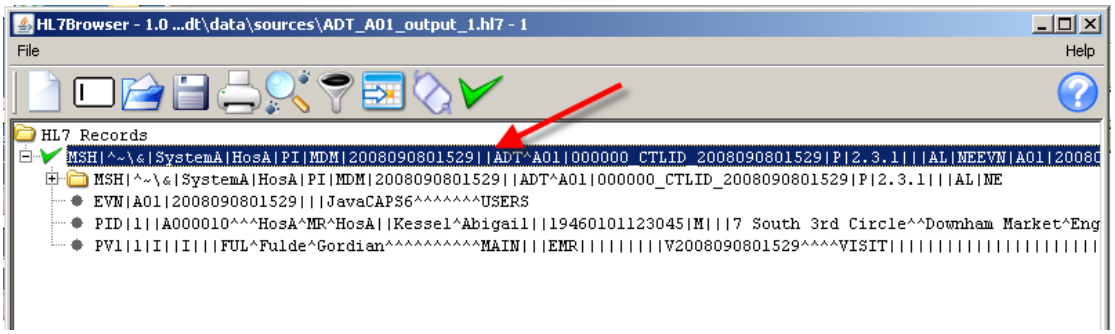
Right-click on the name of the new shortcut and choose Properties.
Change “Start in” property to “C:\tools\Hl7Browser.1.0”. Change “Run” property to Minimized. Click OK.



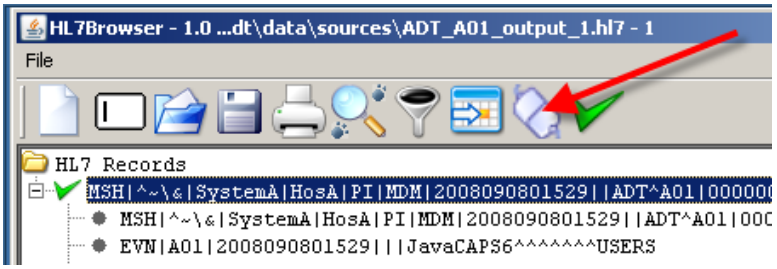
Double-click the HL7 Browser shortcut to start the browser and confirm that the shortcut works. Click on the “Open an HL7 File” button, navigate to c:\hl7\adt\data\sources and select the ADT_A01_output_1.hl7 file.



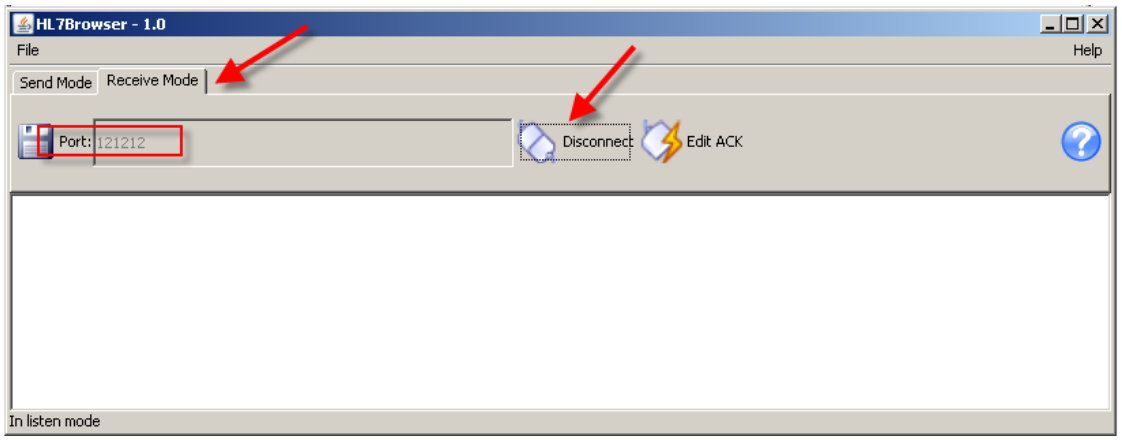
Double-click on the message text to expand it and see the next level of details.



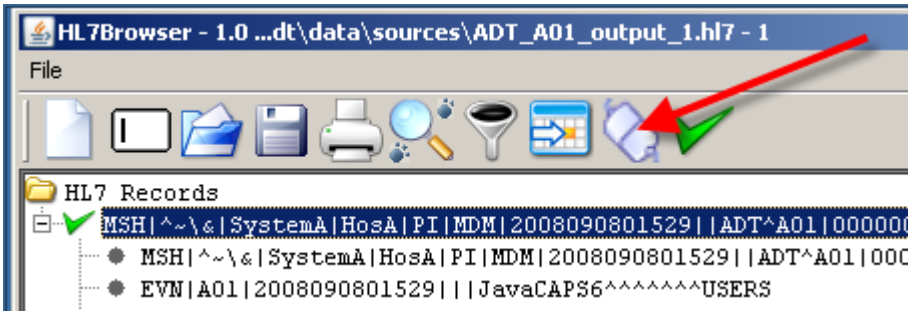
Click “run the network utility” button in the HL7 Browser.



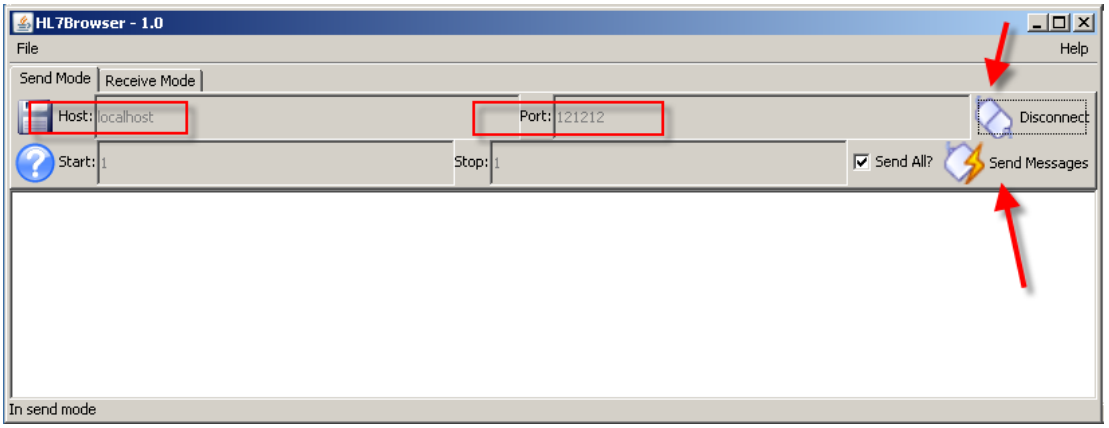
Click on the Receiver Mode tab, enter 12121 for port number and click Connect (which changes to Disconnect when connection is active).



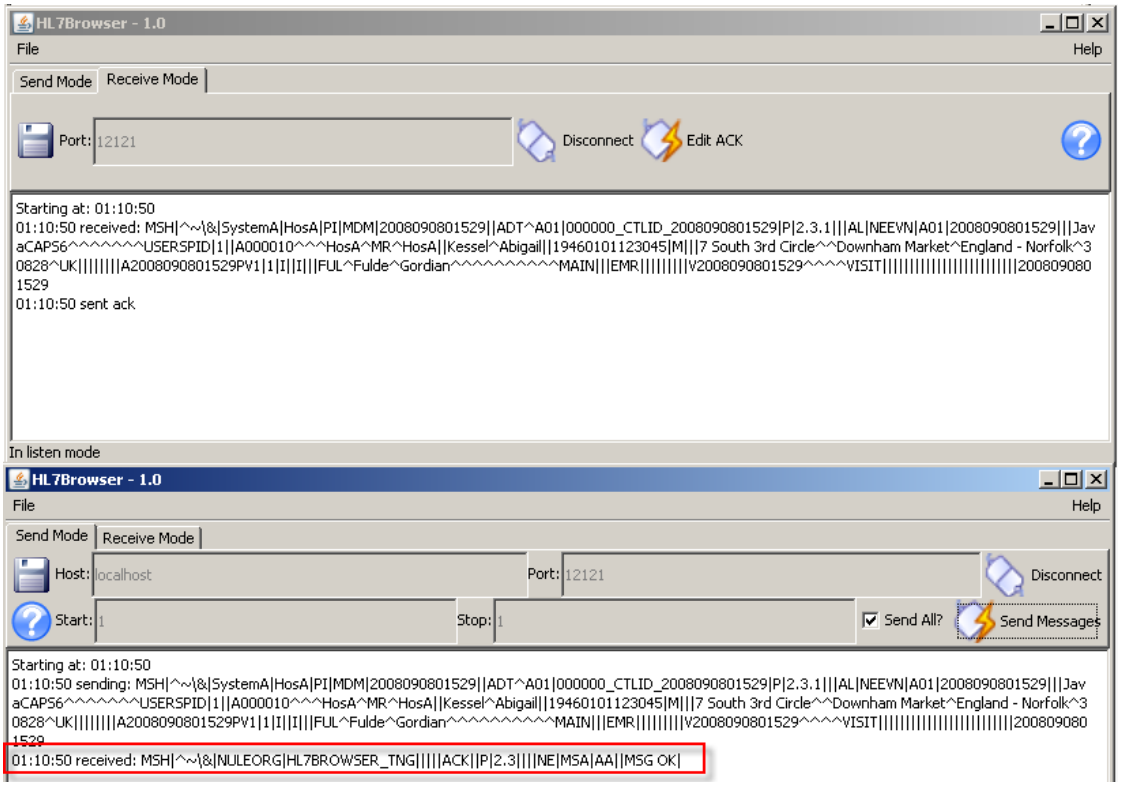
Click “run the network utility” button in the HL7 Browser again.



Enter localhost and 12121 for Host and Port respectively. Click Connect and click Send Messages.



Note message exchange in the sender and the receiver, including a canned ACK in the sender window.



The tool can be used to send messages to another application and receive messages from another application and acknowledge them with a canned ACK message.

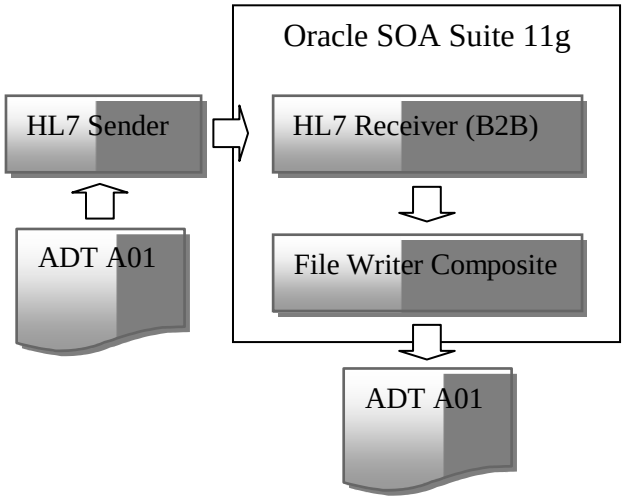
We will use the tool later to look at and send HL7 messages.

HL7 v2 Receiver Solution

Oracle uses the SOA Suite B2B component to provide HL7 v2 messaging support. It uses HL7 v2 message libraries and the B2B engine to provide message parsing and transformation between the native format (HL7 v2 delimited) and XML, which is used internally by the SOAP Suite.
Oracle B2b User's Guide can be found at
http://download.oracle.com/docs/cd/E15523_01/integration.1111/e10229.pdf.

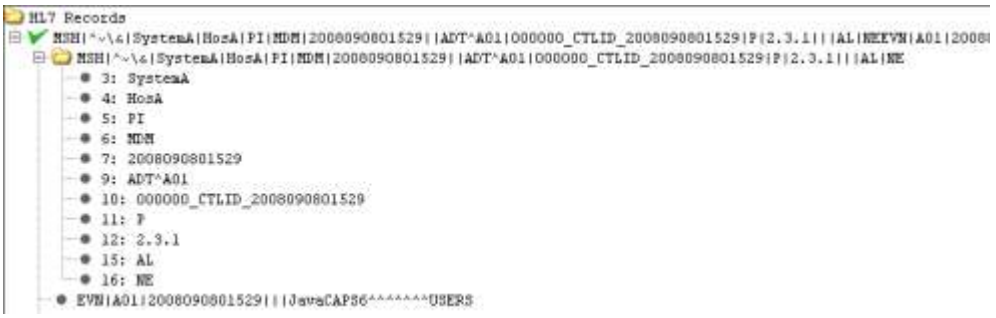
The solution we will be building is a HL7 Receiver, which will receive v2 ADT A01 messages and will write them to files in the file system.
This is the simplest HL7 solution possible.

The solution will consists of a B2B Listener Channel, to which HL7 v2 ADT A01 messages will be sent, and a SOA Composite which will receive these messages and will write them to a file in the file system.



Messages in the sample message set use the following identifiers:

Sending Application	SystemA
Sending Facility	HosA
Receiving Application	PI
Receiving Facility	MDM



Develop HL7 v2 Inbound Solution

To develop the HL7 v2 inbound solution we need to go through a number of steps. The steps are:

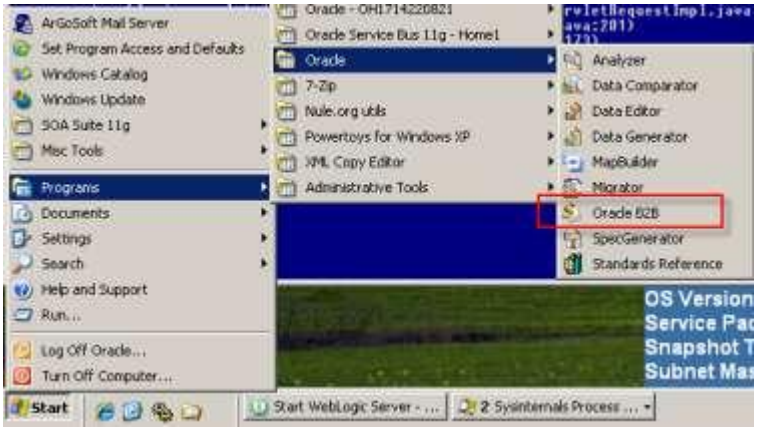
1. Extract HL7 message structures from the standard library
2. Configure Local side of the B2B Partnership
3. Configure Remote side of the B2B Partnership
4. Create and deploy a Partnership Agreement
5. Create and deploy a HL7 Receiver B2B Composite

Extract HL7 Message Structures

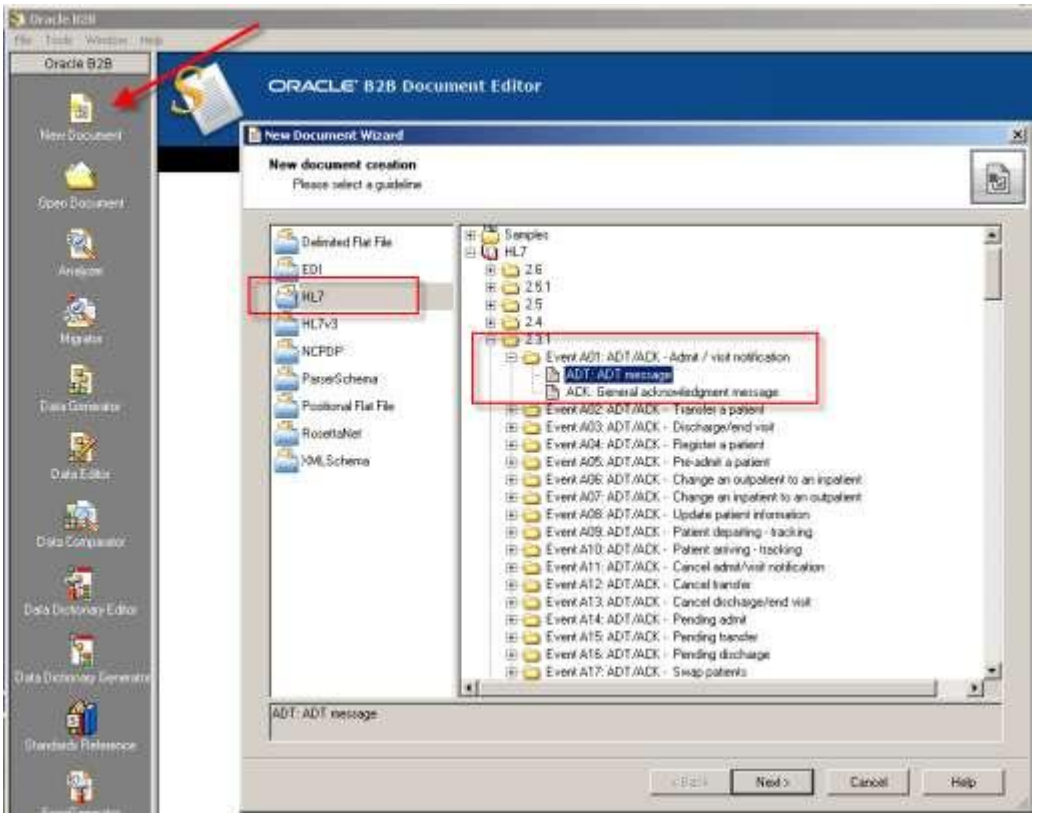
As part of the B2B Document Editor installation we installed a number of standards libraries. One of these libraries was the HL7 v2 library.

Since we will be receiving the ADT A01 we must obtain the eternal representations of the ADT A01 from the Standards Library.

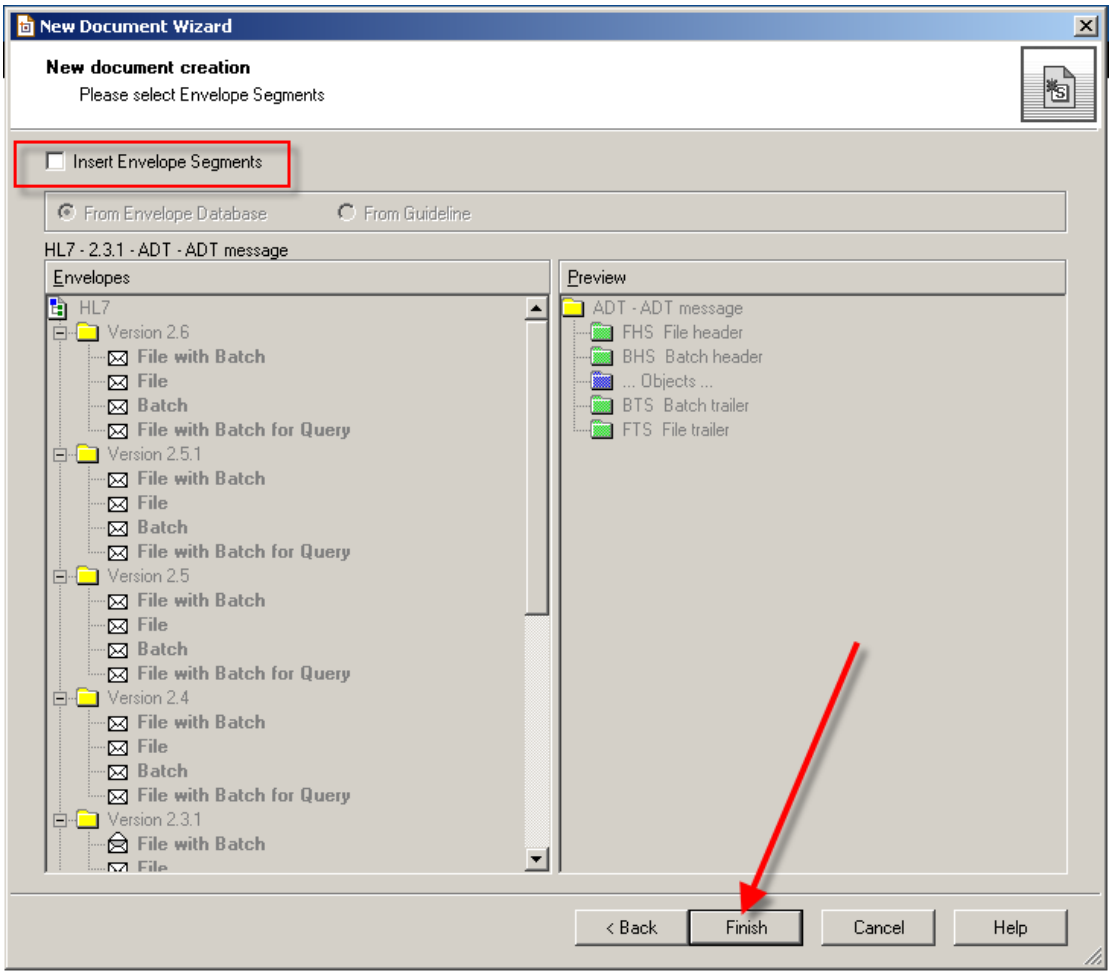
Start the B2B Document Editor, which should be accessible through Start menu → Programs → Oracle as Oracle B2B.



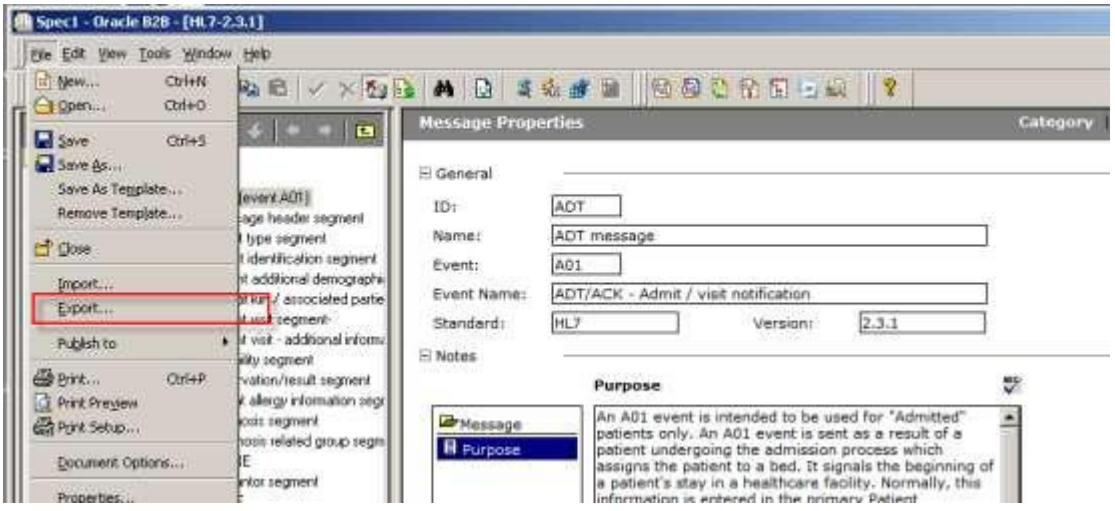
Click the “New Document” button, select HL7 from the list, expand 2.3.1, select Event A01and click Next.



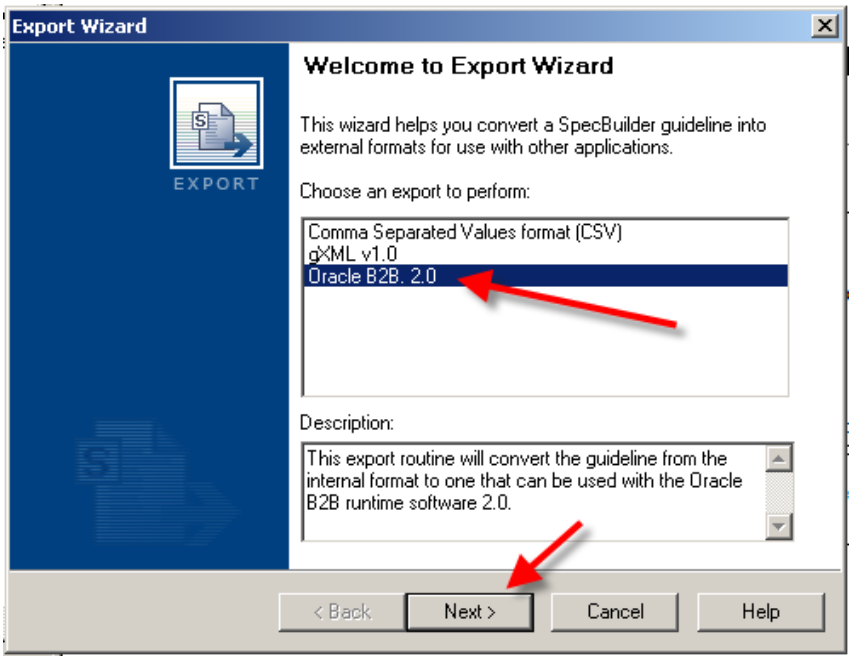
Leave the “Insert Envelope Segments” unchecked and click Finish.



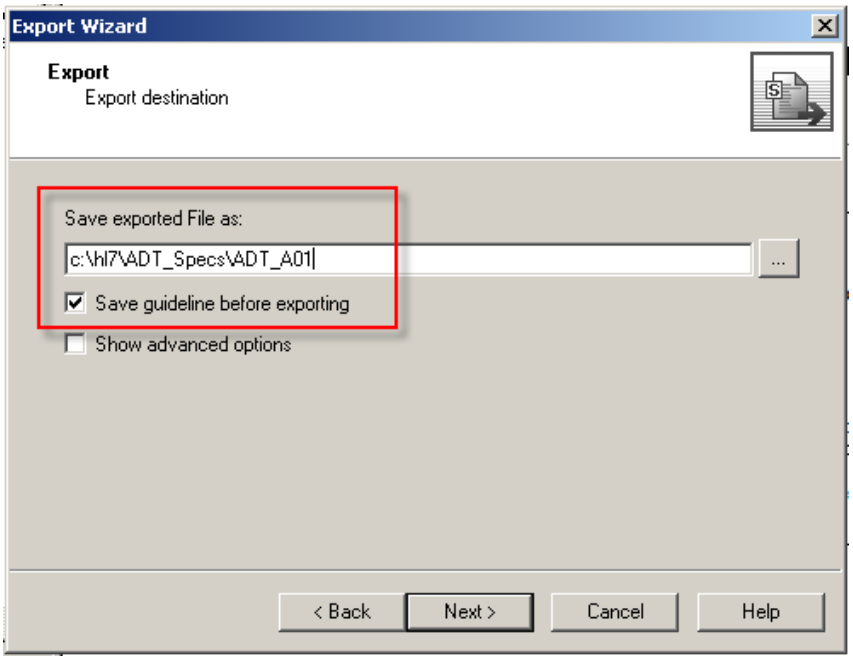
In the new window pull down the File menu and select “Export …”.



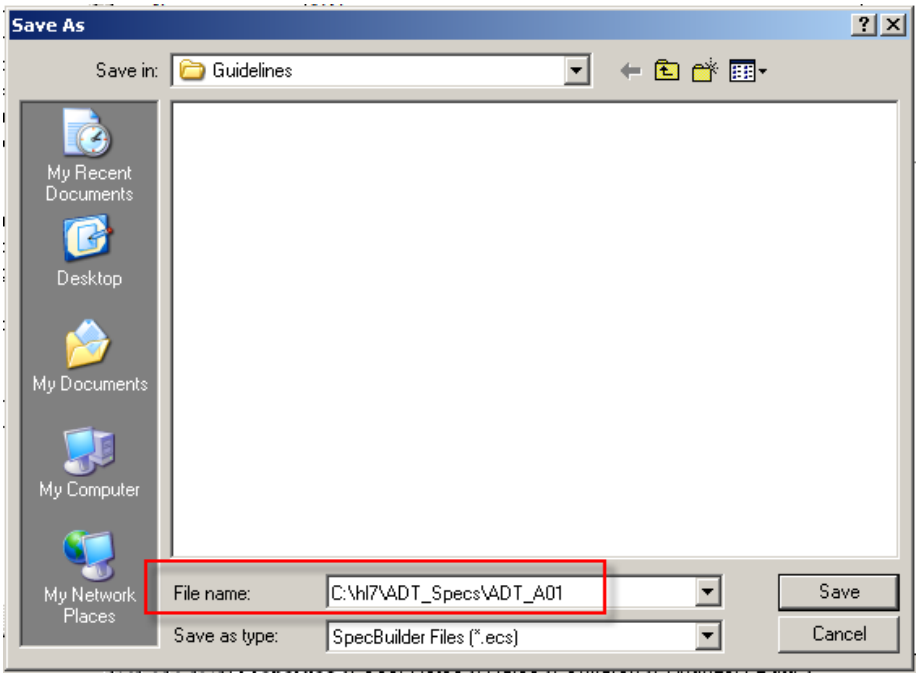
Select Oracle B2B 2.0 and click Next.



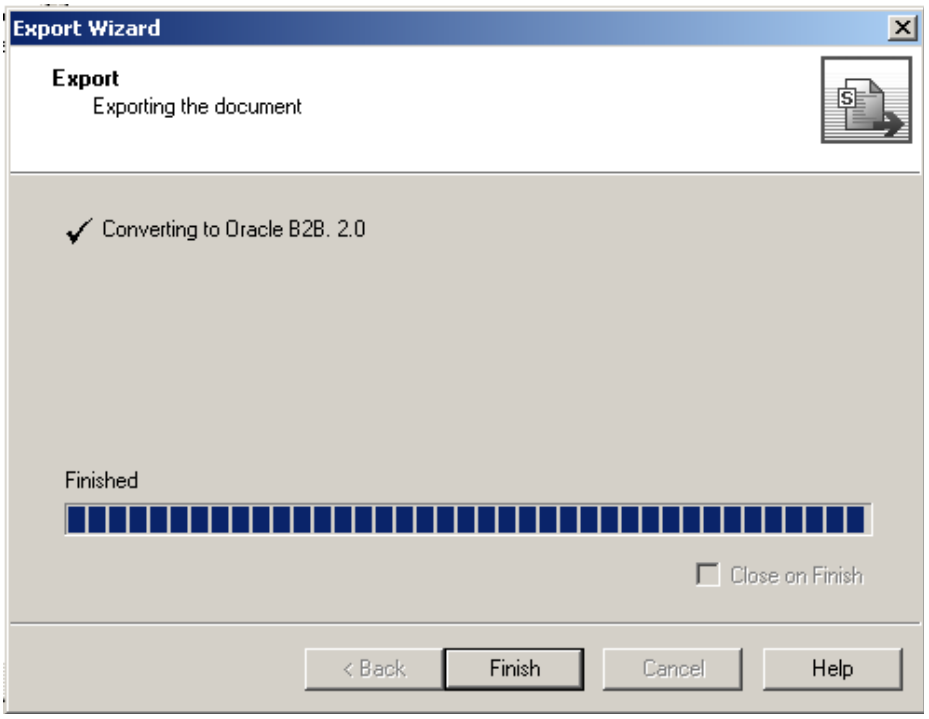
Enter C:\hl7\ADT_Specs\ADT_A01 into the "Save exported File as:" entry field, check the "Save guideline before exporting" checkbox and click Next.



Enter C:\hl7\ADT_Specs\ADT_A01 into the "File name:" text box and click Save.



Click Finish.



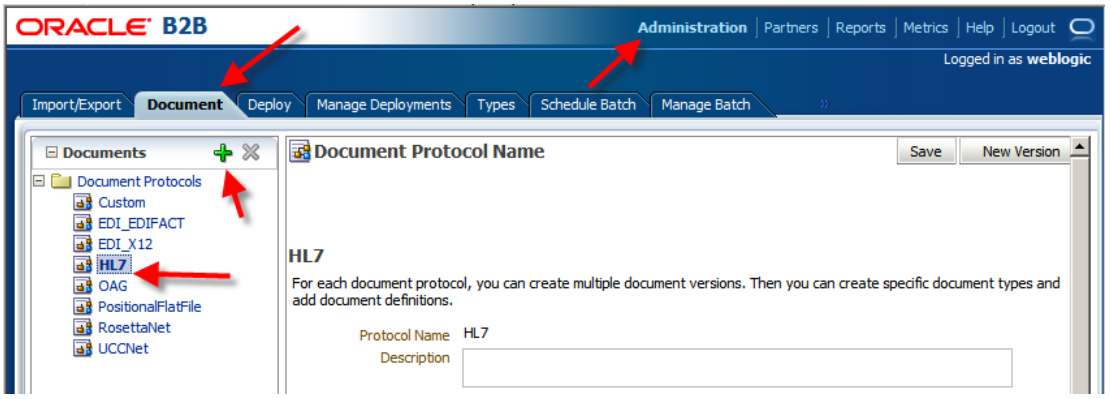
Close both windows. We are done with the B2B Document Editor for this exercise.

We exported the HL7 v2 ADT A01 definition in both the XML Schema form and the format used by the EDIFECS engine. The former can and will be used in the SOA Composite and the later is required to configure the B2B document for the trading partnership we will create shortly, and is used at runtime by the EDIFECS Engine.

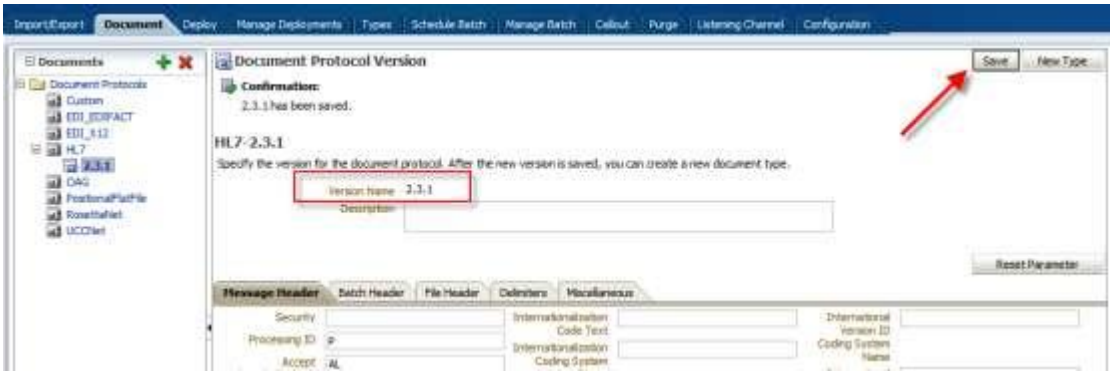
Configure B2B Partnership

Start the B2B Trading Partner Manager console by pointing the web browser at http://localhost:7001/b2b. Log in as weblogic/welcome1.

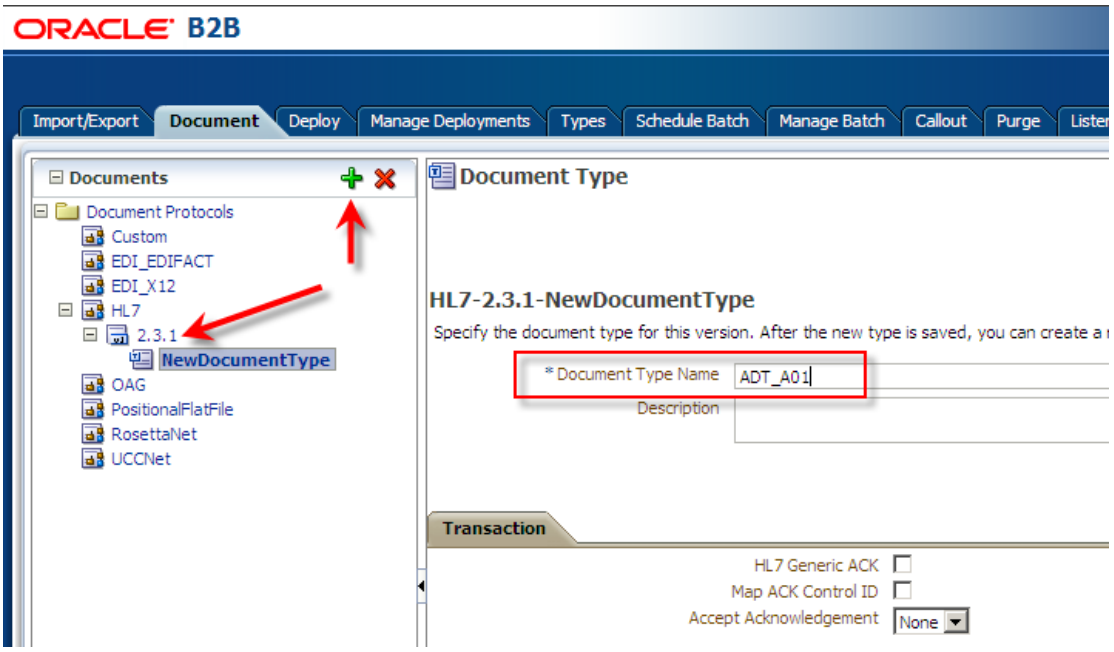
Click the Administration link, select the Document Tab, select HL7 in the node tree and click the large Plus sign to add a new document.



Enter 2.3.1 for Version Name and click Save. The screenshot shows the state after the Save button has been clicked.

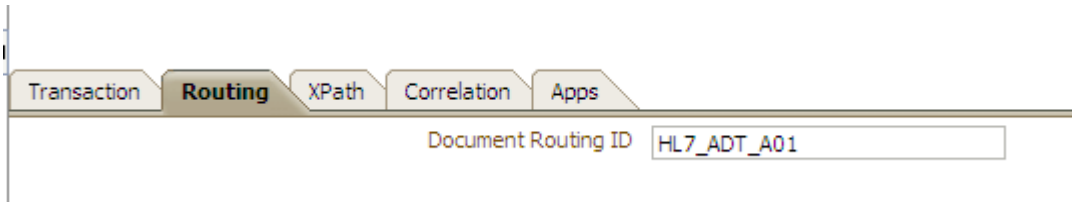


With the new node, 2.3.1, selected, click the large Plus sign to create a new document under that node, name if ADT_A01 and click Save. This name is derived from the Message Type and Trigger Event in the HL7 message.



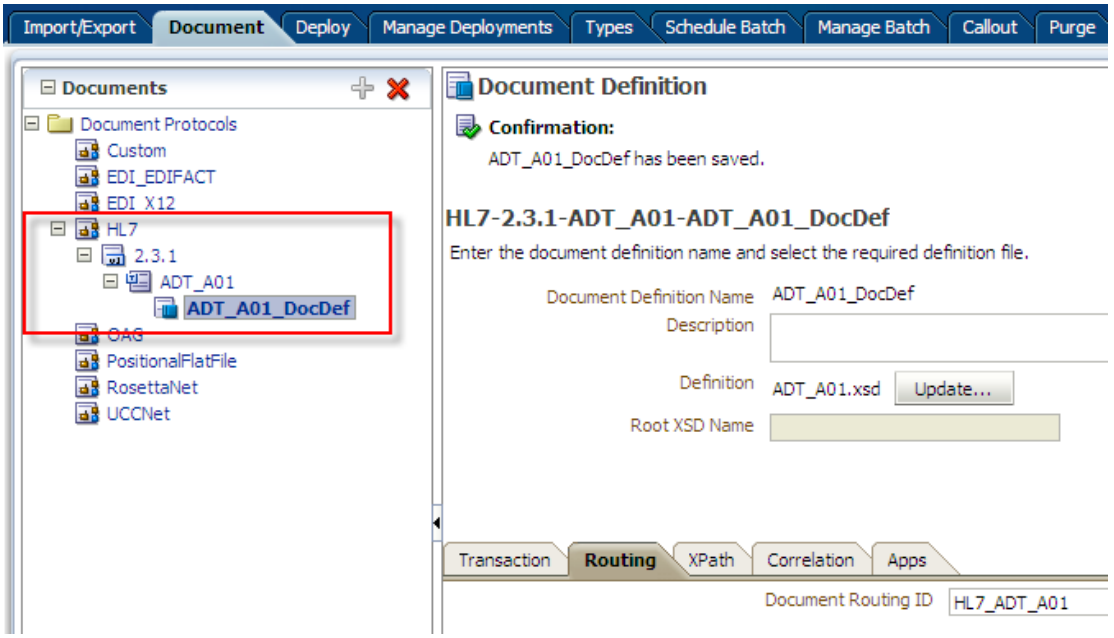
With the new node, ADT_A01, selected click the large Plus sign to add a new document under that node. Provide the name of ADT_A01_DocDef, locate and choose c:\hl7\ADT_Specs\ADT_A01.xsd file for document definition and c:\hl7\ADT_Specs\ADT_A01.ecs file for Transaction Set ecs File.

Click the Routing Tab and enter HL7_ADT_A01 into the Routing ID field.



Click Save to save the changes.

A hierarchy like that shown below should be now defined.

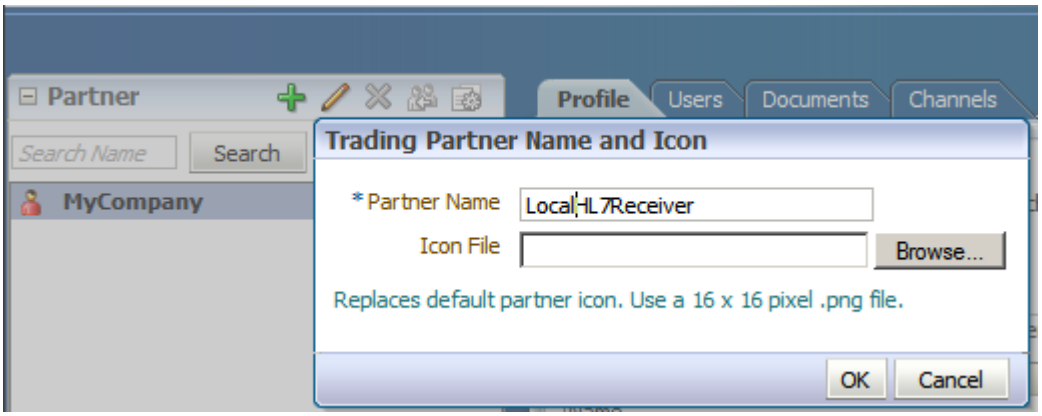


This defines the documents to be used in setting up message exchanges.

Now we need to configure trading partners, the local partner and the remote partner.

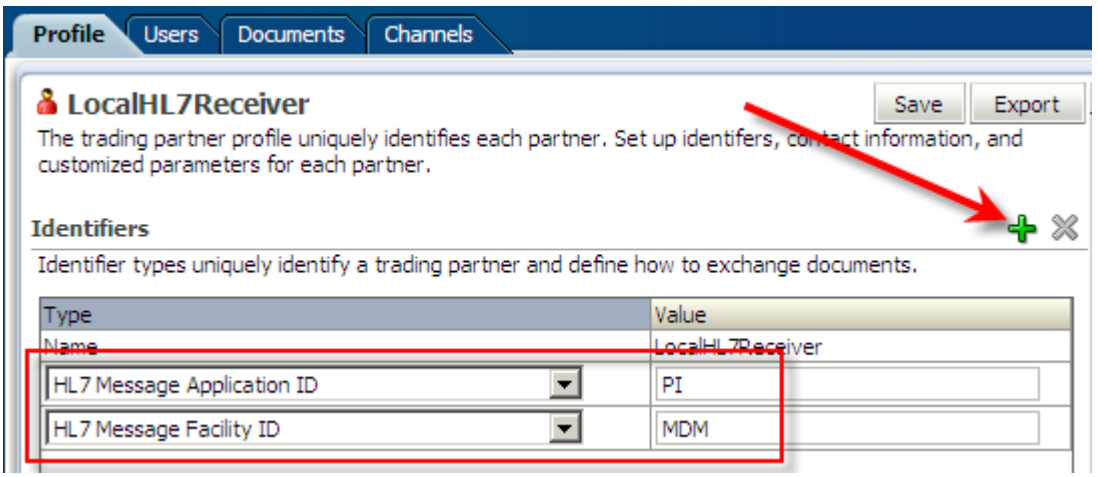
Click the Partners navigational link, select the default pre-defined MyCompany partner and click the Edit button (Pencil icon).

Change the name to LocalHL7Receiver and click OK.



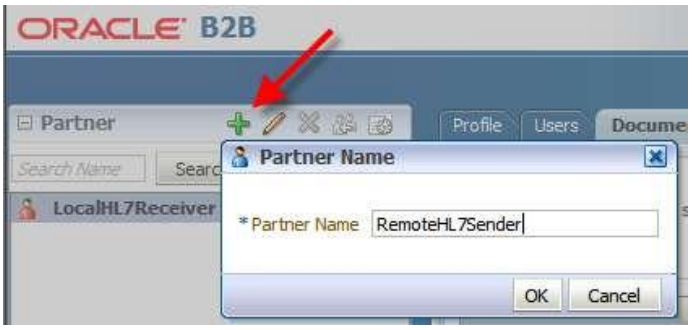
Click the large Plus sign in the right hand window to add two identifiers. From the drop-down lists choose HL7 Message Application ID and set it to the value of PI for one and HL7 Message Facility ID set to the value of MDM for the other. Click Save.

These are the IDs of the receiver side. This is what the messages are expected to carry in MSH-5 and MSH-6 fields.

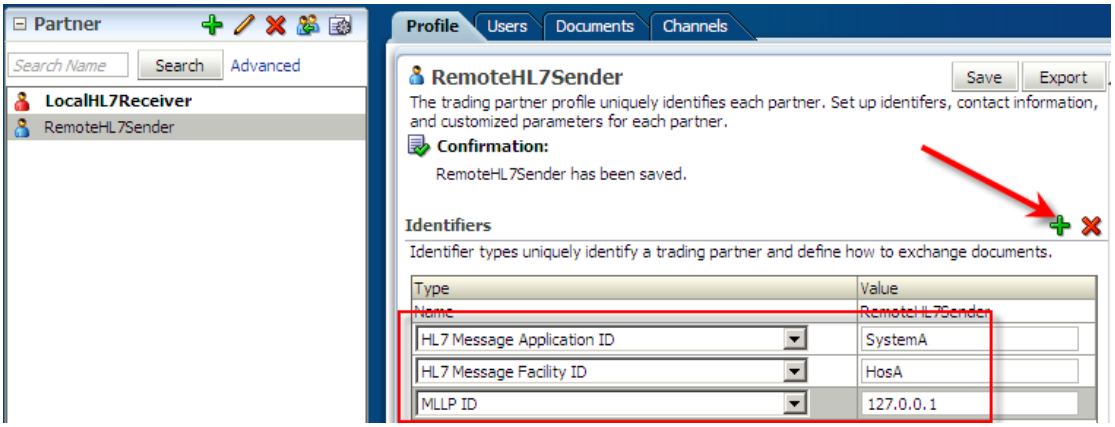


With the LocalHL7Receiver selected click the Documents Tab, uncheck the Sender checkbox and click Save. We are configuring the HL7 inbound so the local role is that of a receiver of messages.

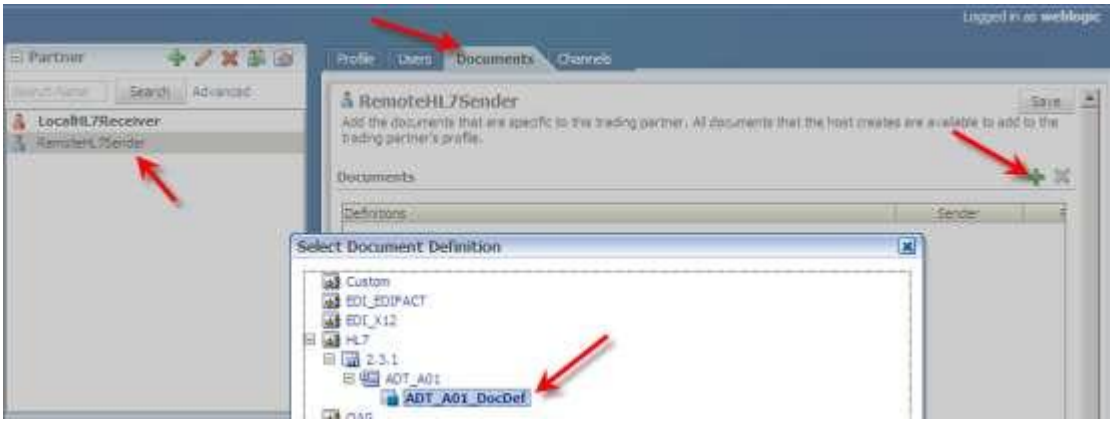
Click the large Plus sign to create a new trading partner, named RemoteHL7Sender, enter the name and click OK.



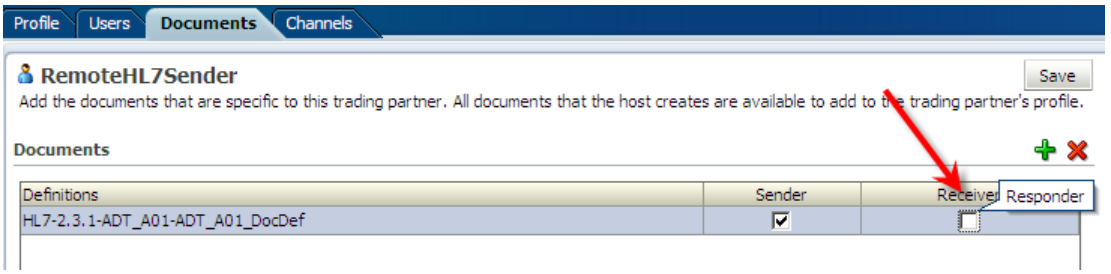
With the RemoteHL7Sender selected click the large Plus sign in the right hand panel to add three identifiers, HL7 Message Application ID set to the value of SystemA, HL7 Message facility ID set to the value of HosA and MLLP ID set to the value of 127.0.0.1. Click Save.



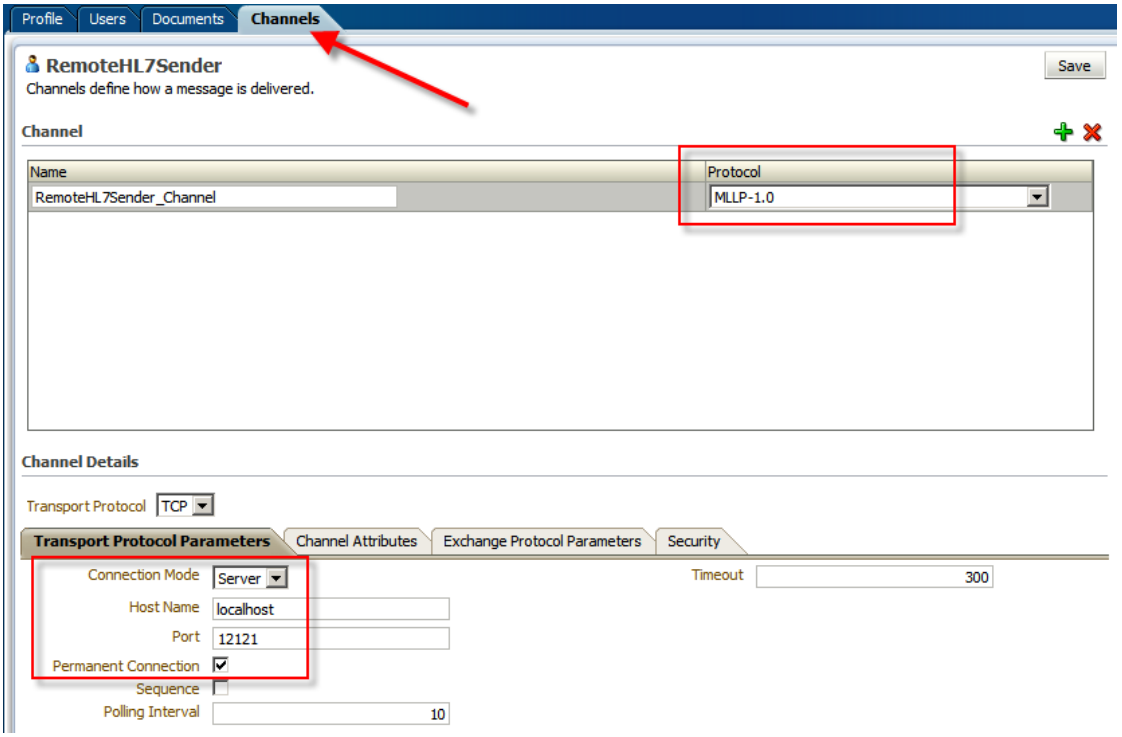
With the RemoteHL7Sender selected, click the Documents Tab, click the large Plus sign to add a document, select HL7 → 2.3.1 → ADT_A01 → ADT_A01_DocDef and click Add.



Uncheck the Receiver checkbox and click Save.



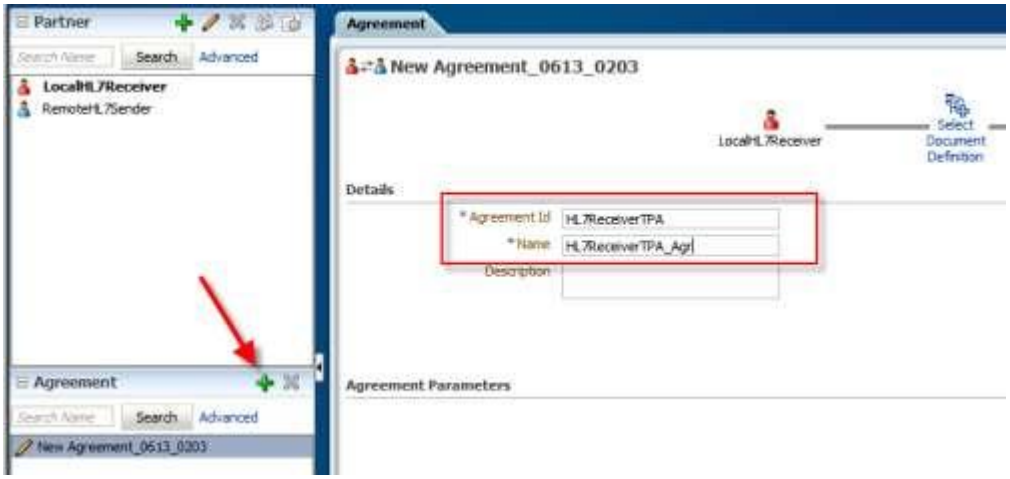
With the RemoteHL7Sender partner selected click the Channels Tab, select MLLP-1.0 from the Protocol drop-down, configure Connection Mode: server, Host Name: localhost, Port: 12121 properties and check the Permanent Connection checkbox.



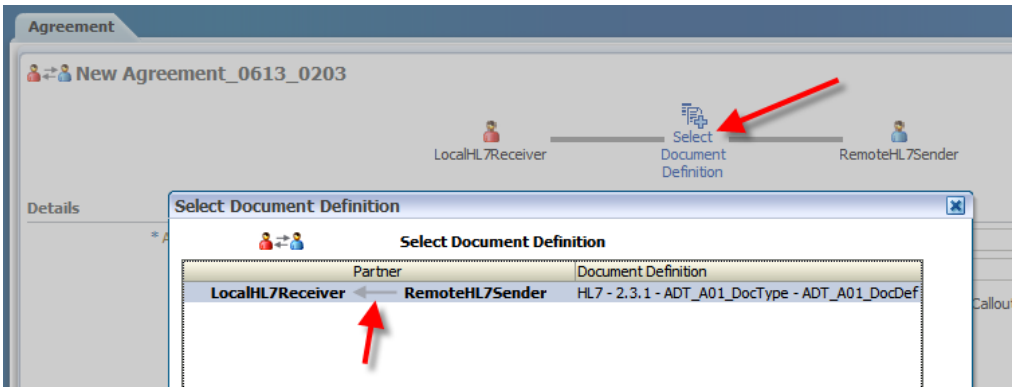
In the Channel Details section of the panel click the Exchange Protocol Parameters Tab. Choose Default from the Immediate ACK drop-down, check the Map ACK Control ID and Map Trigger Event checkboxes. Finish by clicking the Save button.

We defined the document types the remote partner will be sending and configured the listener we need to use on our side to receive messages from that partner.

Click the large Plus sign in the Agreements section to add a new trading partnership agreement. Provide the Agreement ID of HL7ReceiverTPA and the Name of HL7ReceiverTPA_Agr.

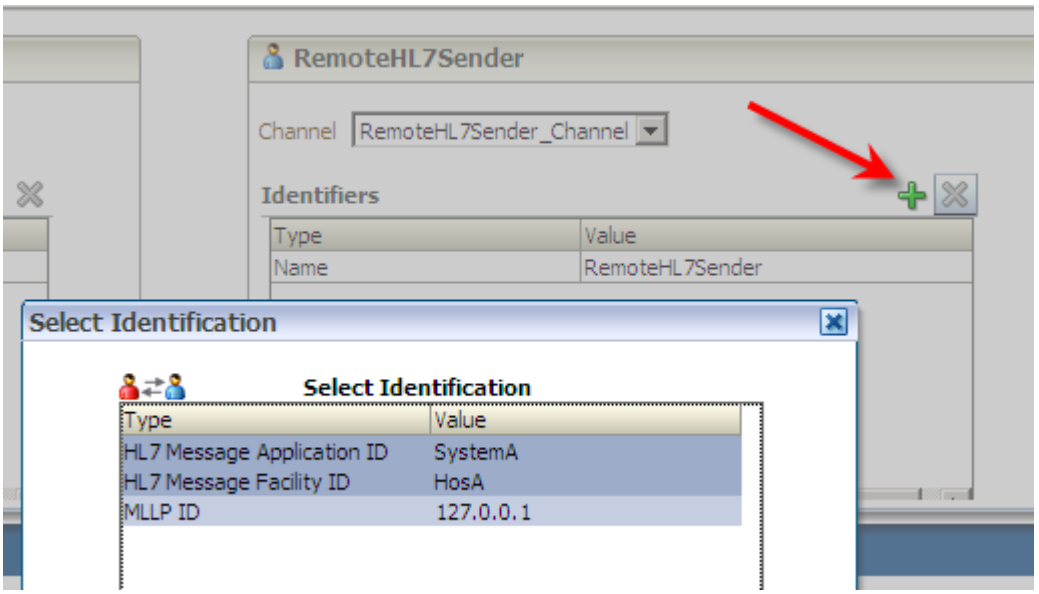


Click the "Select Document Definition" icon in the top centre of the panel, select the definition of the document being sent from the remote sender to local receiver and click OK.

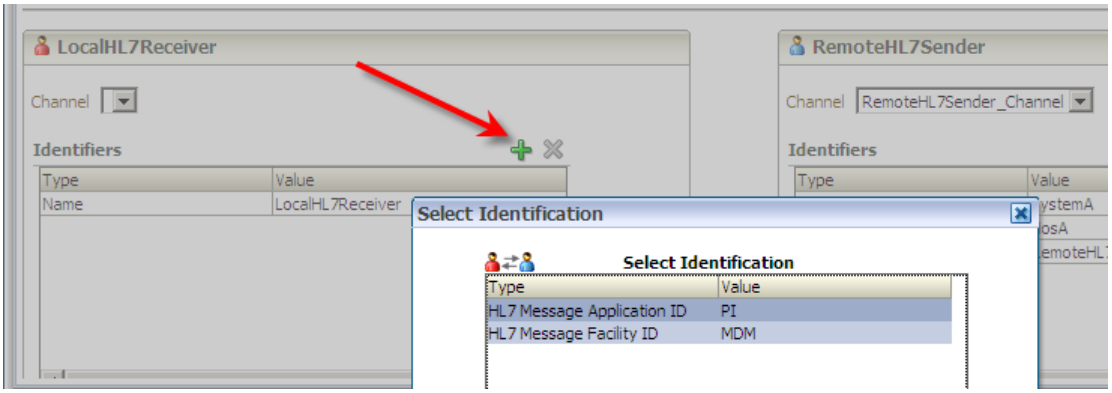


Choose the RemoteHL7Sender_Channel from the Channel drop-down on the RemoteHL7Sender panel in the bottom right portion of the display.

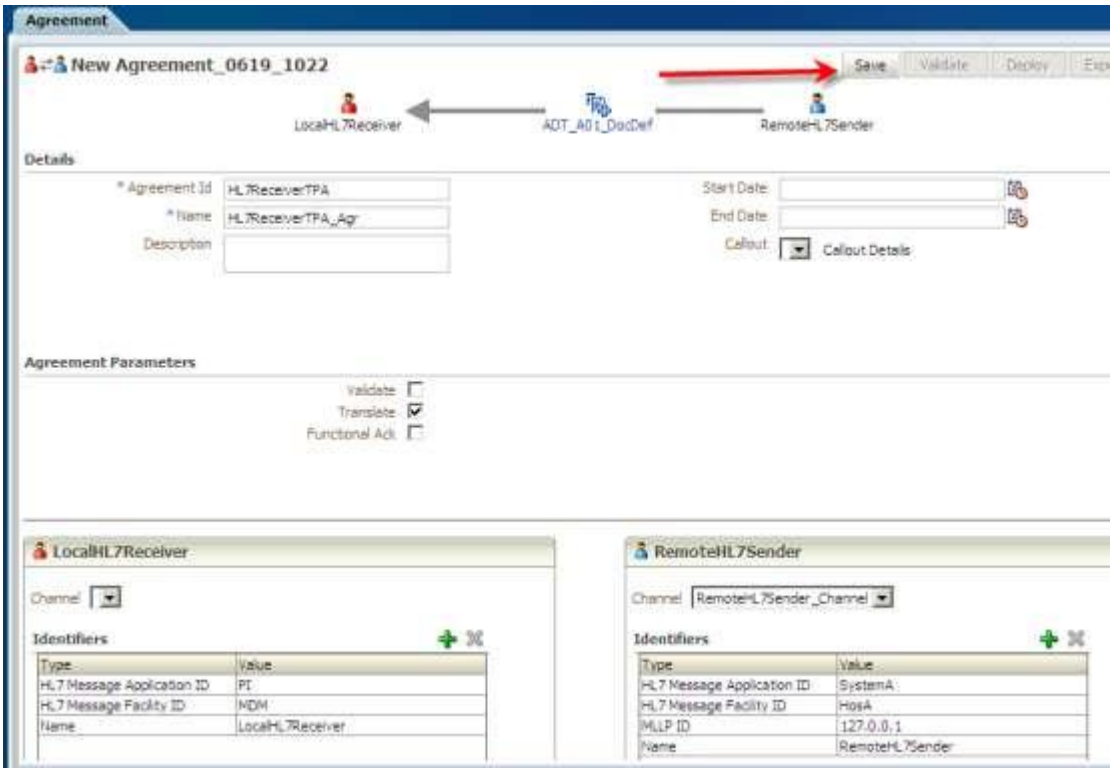
In the bottom right panel click the large Plus sign to specify identifiers defined when the RemoteHL7Sender partner was created.



In the bottom left panel click the large Plus sign to add identifiers used by the LocalHL7Receiver partner.



Click Save to save the new trading partnership agreement.

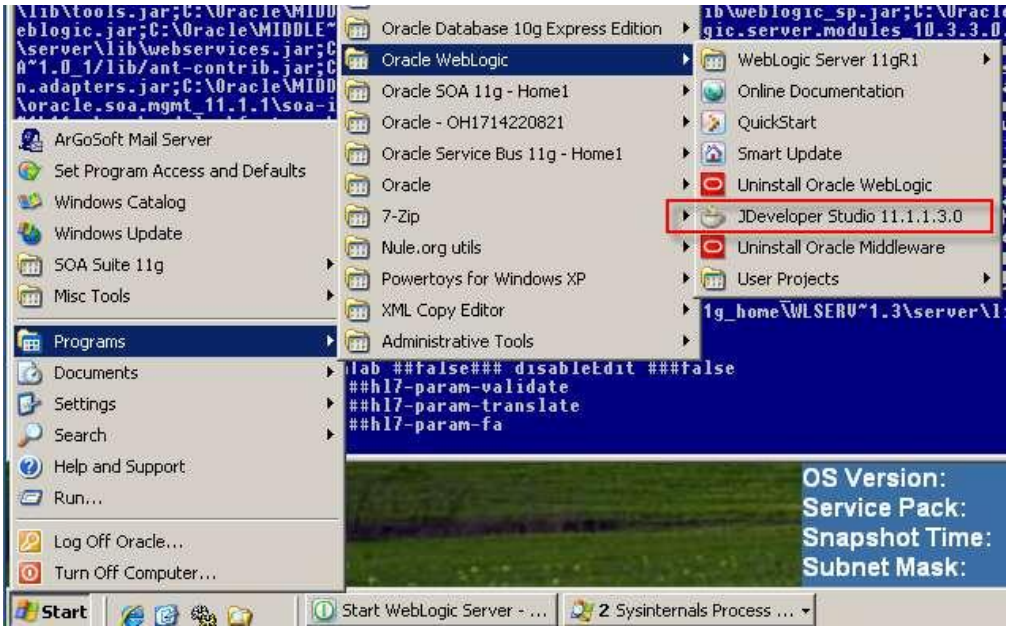


Click Deploy link to deploy the trading partnership agreement.

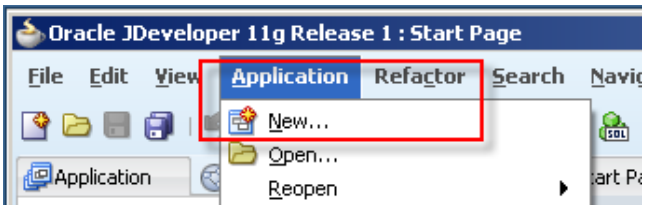
We are finished with the trading partner management console for the time being. The partnership has been defined and deployed. Now we need the runtime solution that will receive messages, according to the trading partnership agreement, and will write them to a file in the file system. This solution, which will be a SOA Suite Composite, will be developed using the JDeveloper IDE and will be deployed to the WebLogic Application Server.

Develop File Writer Solution

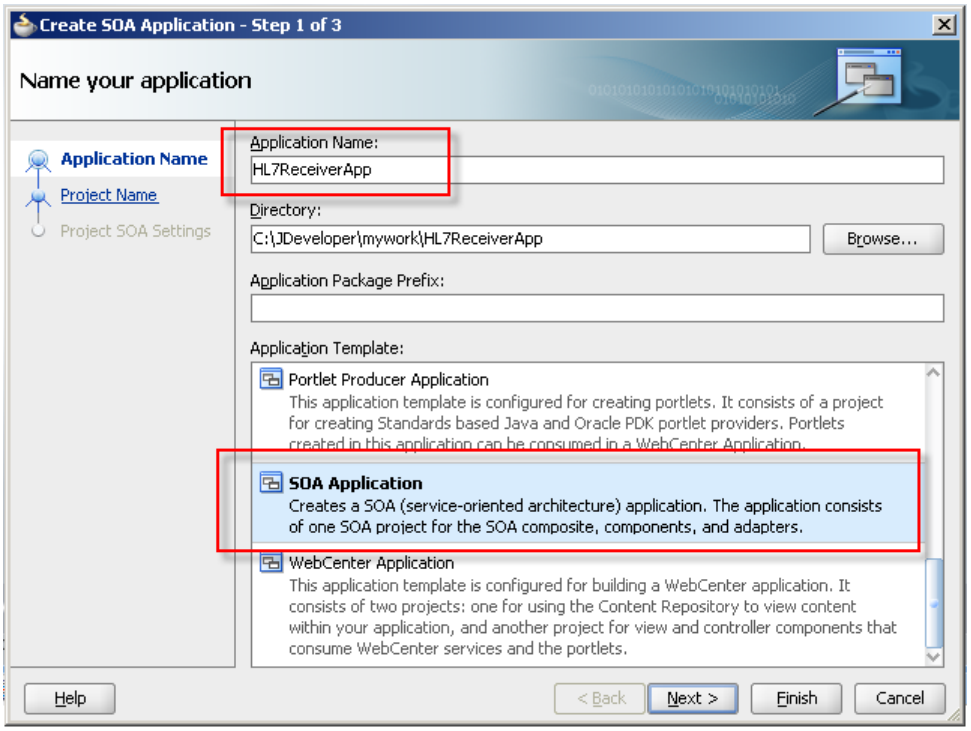
Start JDevelope IDE, perhaps by following Start menu → Programs → Oracle WebLogic → JDeveloper Studio 11.1.1.3.0 path through the menus.



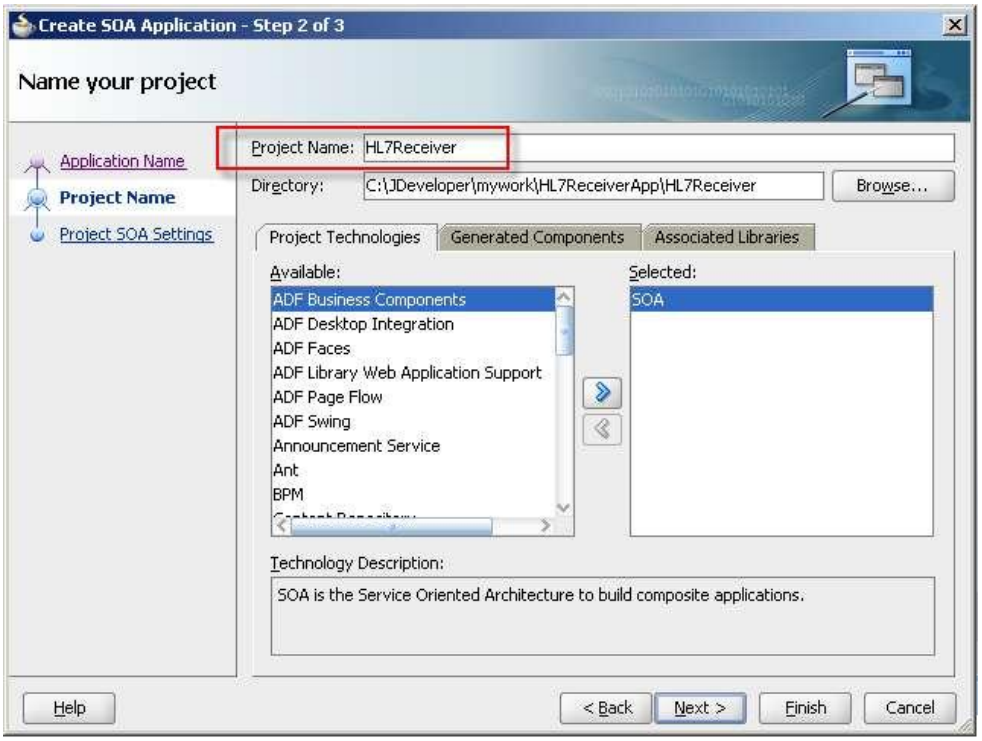
Once JDeveloper is up create a new SOA Application.



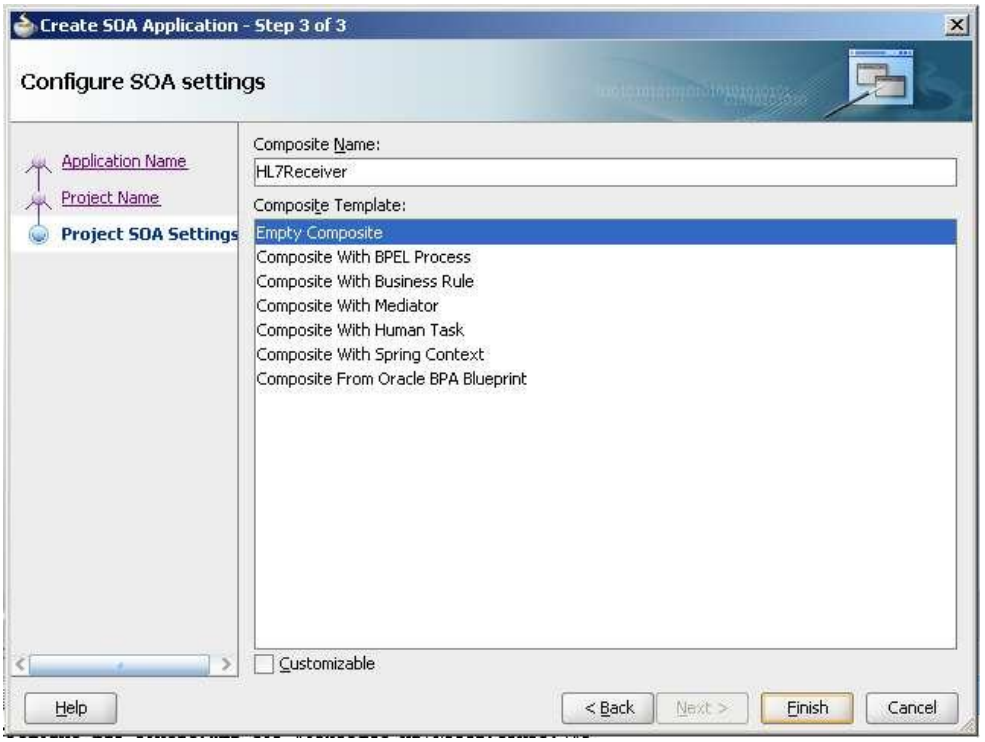
Name this application HL7ReceiverApp, choose SOA Application template and click Next.



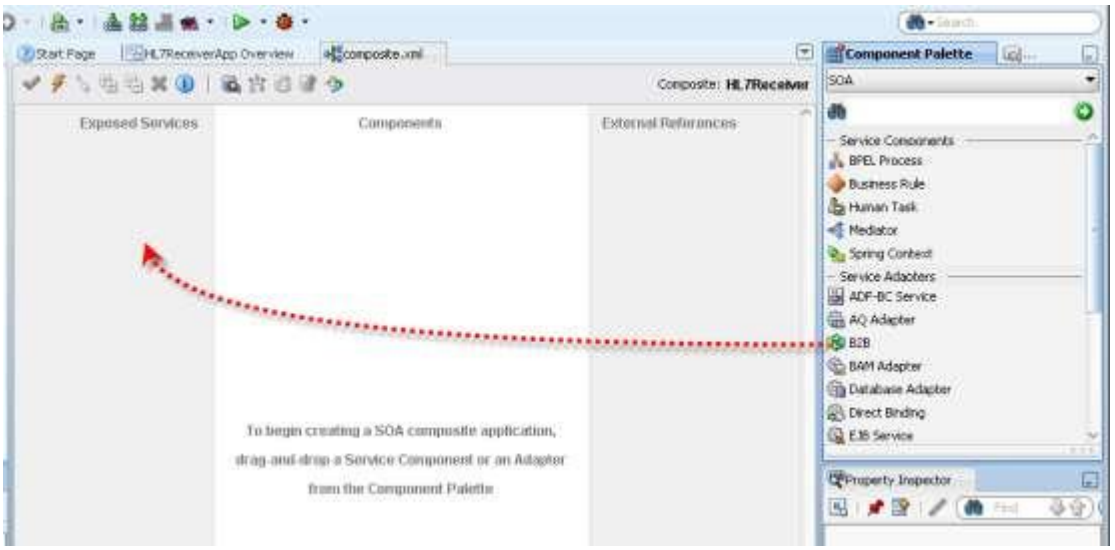
Name the project HL7Receiver and click Next.



Accept the default Composite Name of HL7Receiver and default template Empty Composite, and click Finish.



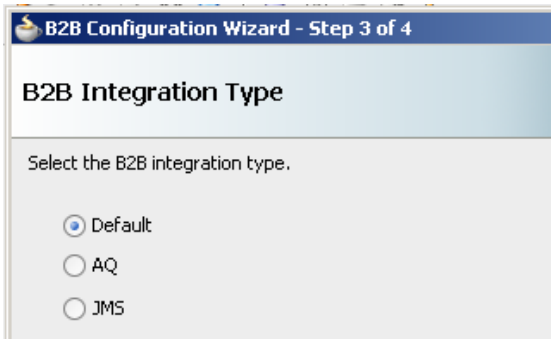
Drag the B2B Service Adapter from the list of service adapters in the SOA components palette.



Click Next to the "Welcome to B2B Configuration Wizard" dialogue window.

Set the Service Name to HL7ReceiverIn and click Next.

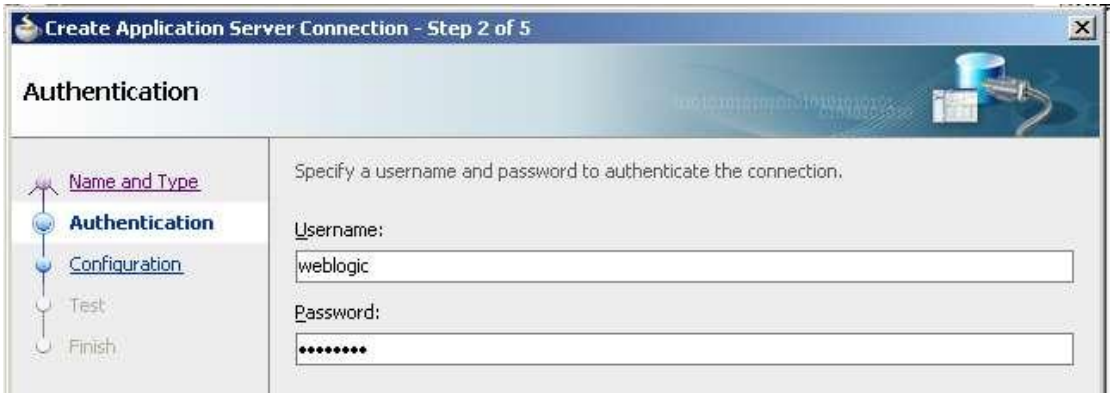
Accept default "Default" and the B2b Integration Type and click Next.



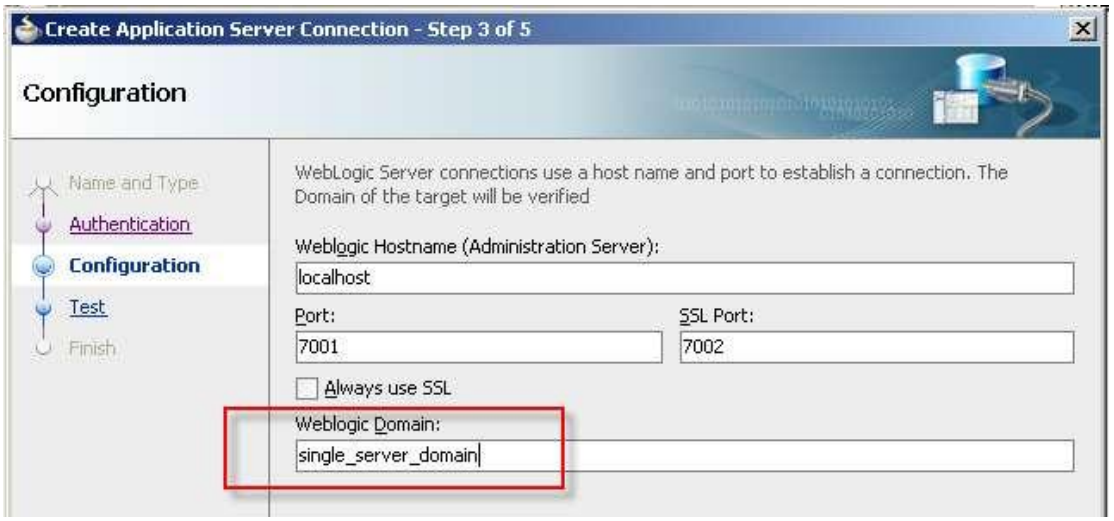
This is the first time we are developing a composite in this installation so there will be no AppServer Connection. Click the large Plus sign next to the AppServe Connection, to start the AppServer connection definition wizard. We will resume the B2B Interface Wizard shortly.

Give the connection the name of WebLogic_Local, Accept the default of WebLogic 10.3 and click Next.

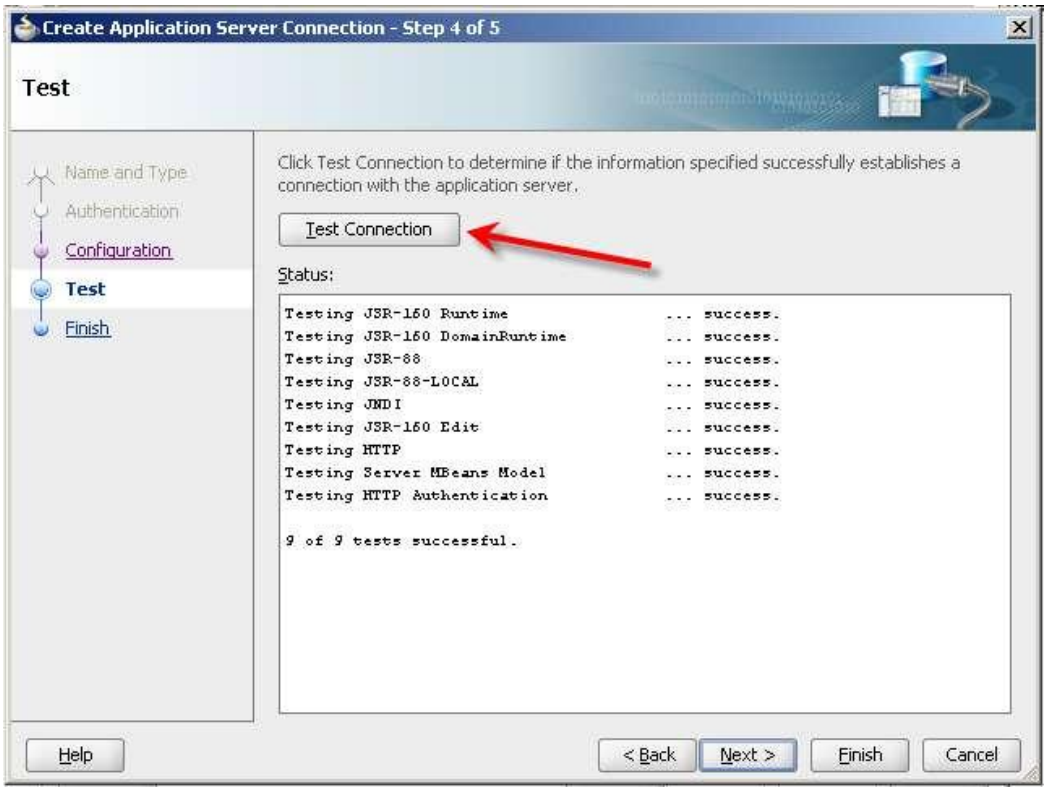
Provide credentials weblogic/welcome1 and click Next.



Set WebLogic Domain to single_server_domain, which is what our domain is called, and click Next.

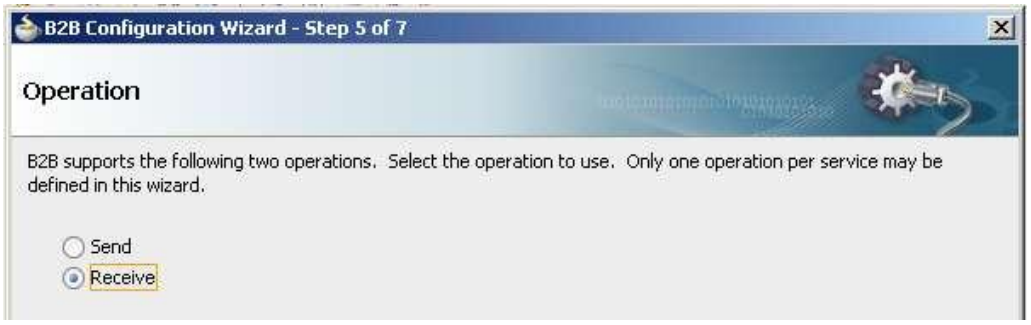


Click "Test Connection" button. Hopefully you will see several Success messages. Click Next and Finish.

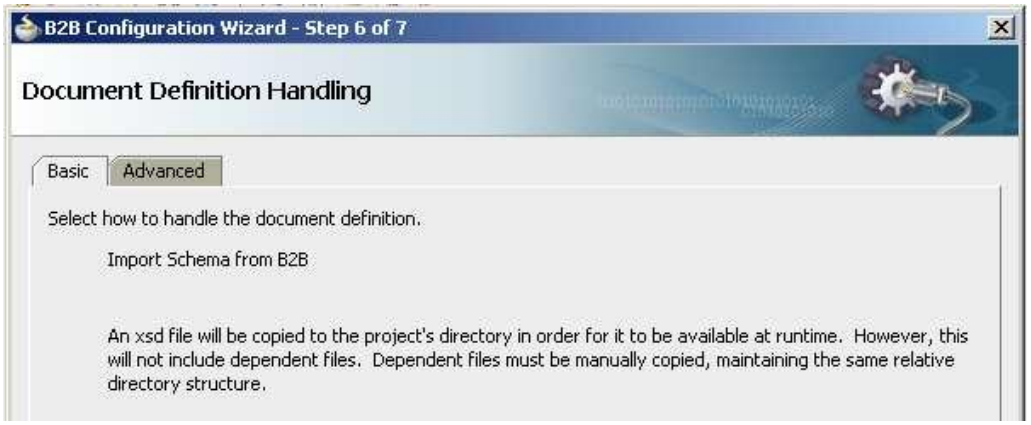


Back in the B2B Interface Wizard the new AppServer Connection, WebLogic_Local, will be selected. Click the Text B2B button. Once the Success dialog opens, click OK, then Next to continue defining the B2B Interface.

Choose the Receive operation and click Next.



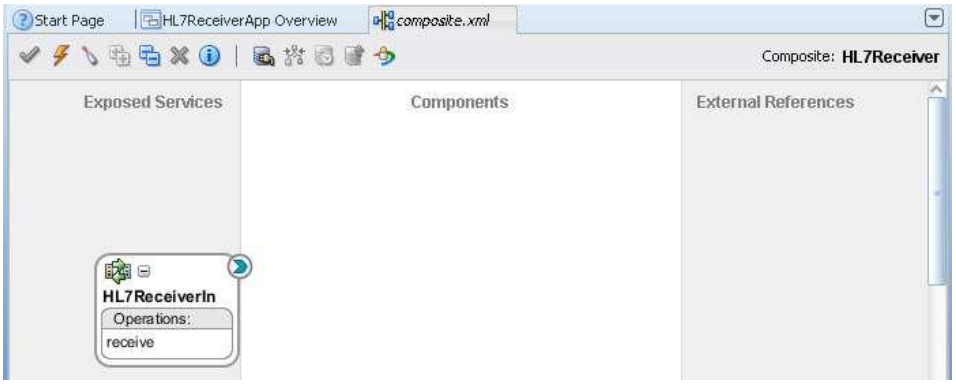
Keep the Document Definition Handling at Basic, the default, and click Next.



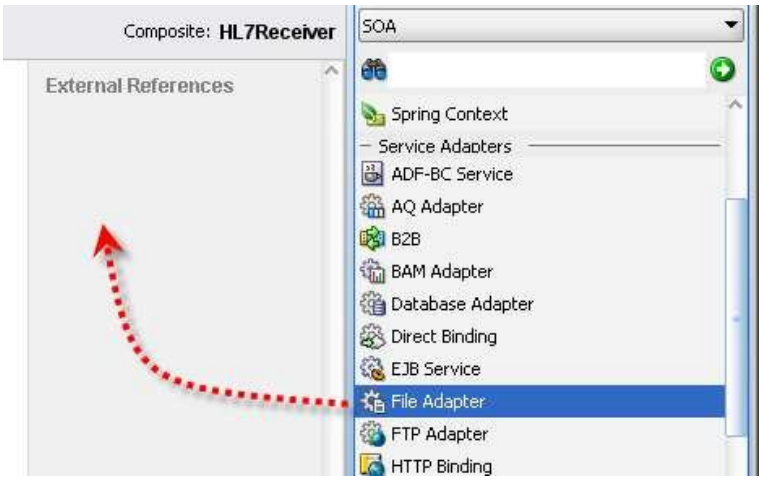
Choose the HL7 2.3.1 ADT A01 document definition and click Next.

Click Finish to complete the wizard.

The HL7ReceiverIn will appear in the composite in the "Exposed Services" swim line.

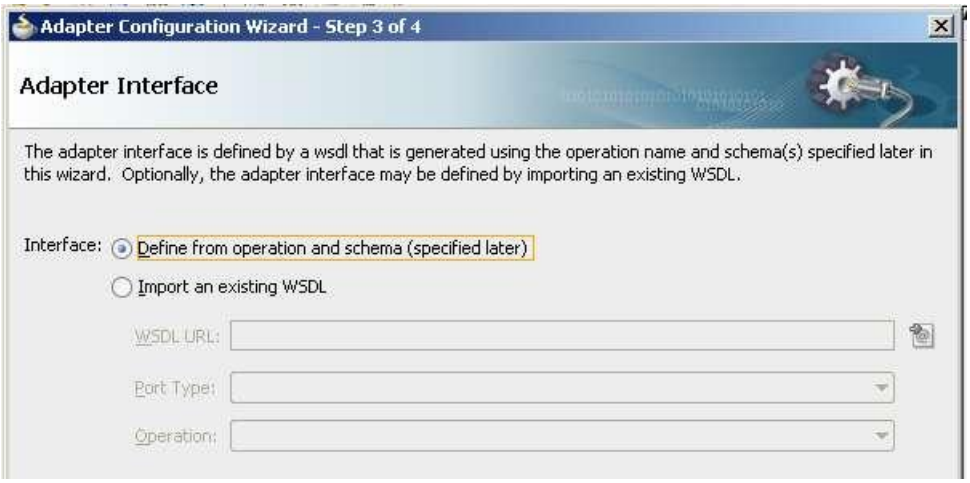


Drag the File Adapter from the Service Adapters component palette to the External References swim line.

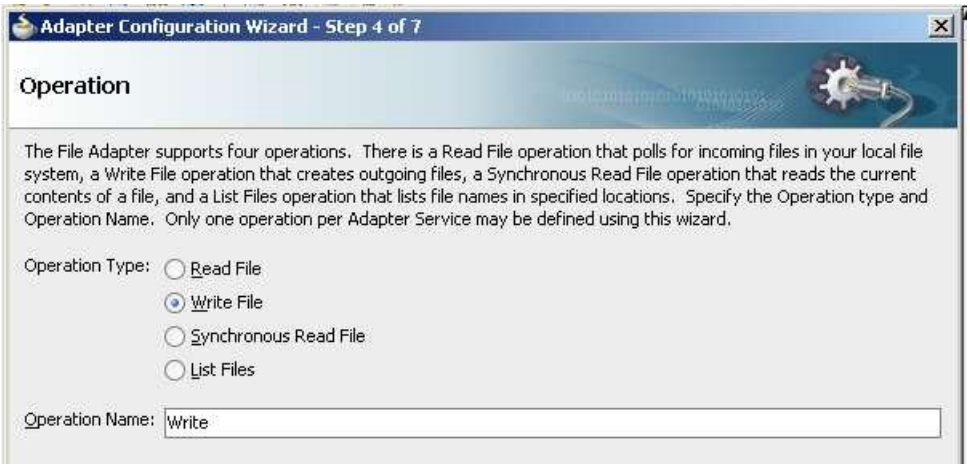


Click Next to the Welcome dialogue, enter HL7FileWriter as service name and click Next.

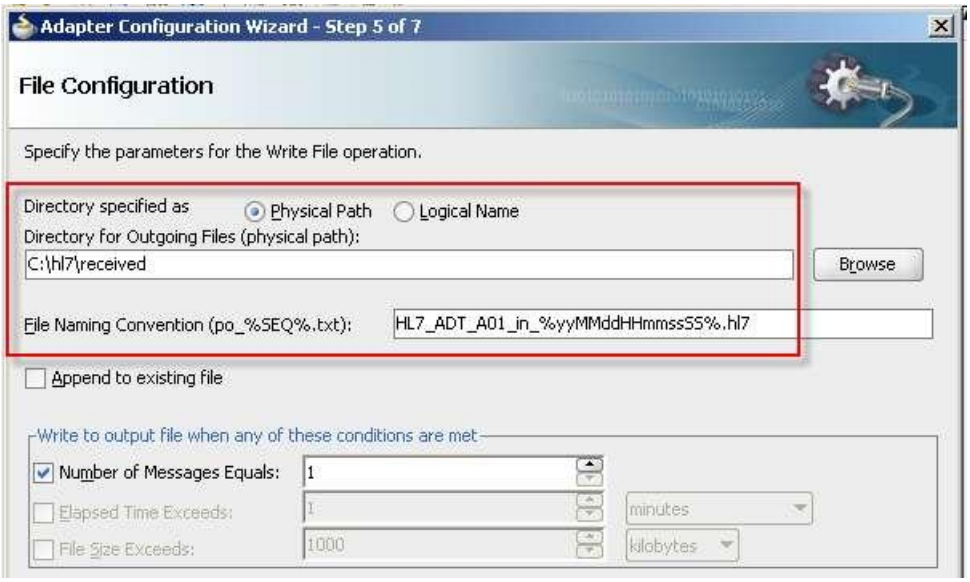
Accept the defaults for Adapter Interface and click Next.



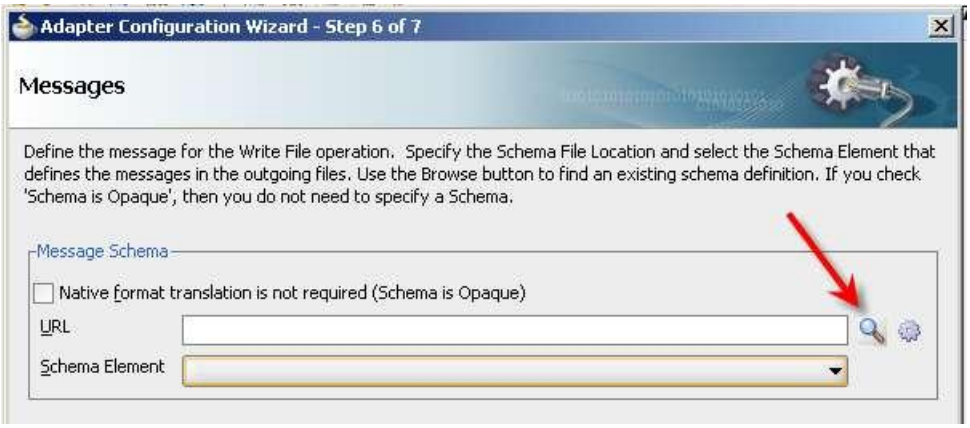
Choose Write File operation type, accept default operation name and click Next.



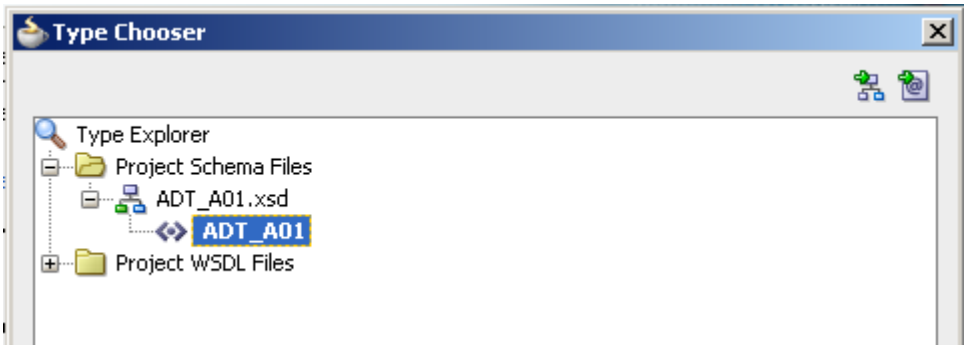
Specify C:\hl7\received as the physical path to the directory where the files will be written. Specify HL7_ADT_A01_in_%yyMMddHHmmssSS%.hl7 as the "file naming convention". Leave all else at defaults and click Next.



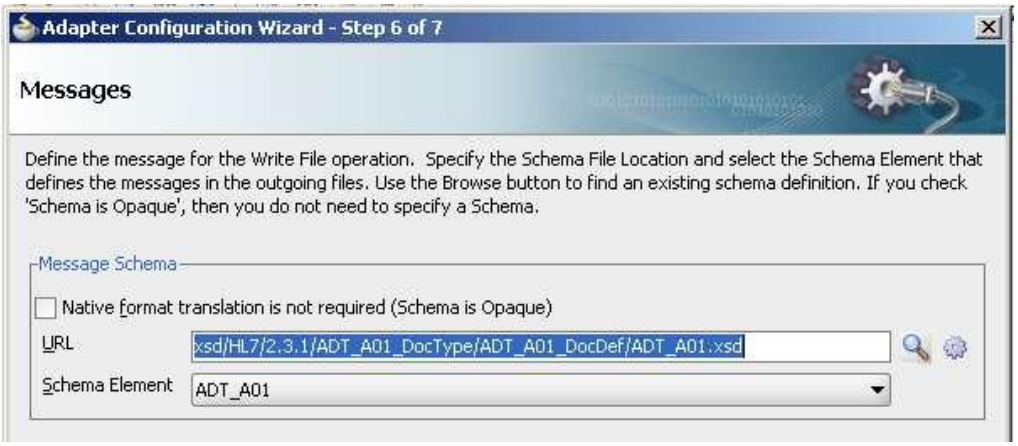
Click “Browse for Schema File”.



In Type Explorer choose ADT_A01 element of the ADT_A01.xsd schema file and click OK.

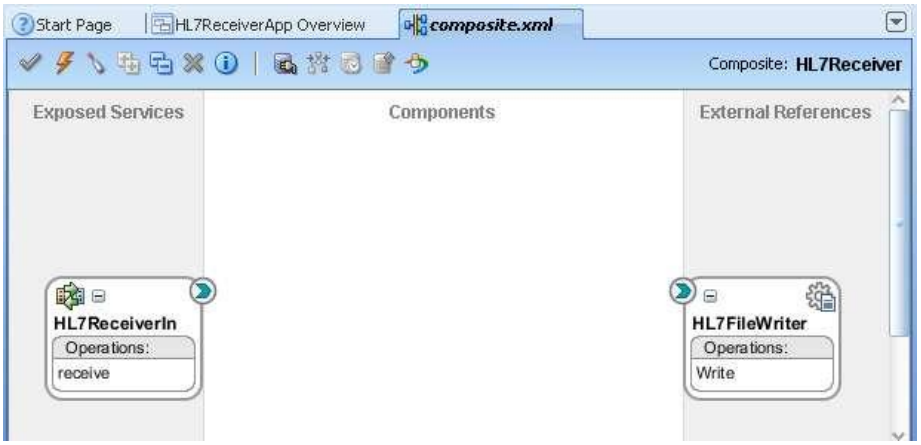


Confirm the schema element, click Next and click Finish.



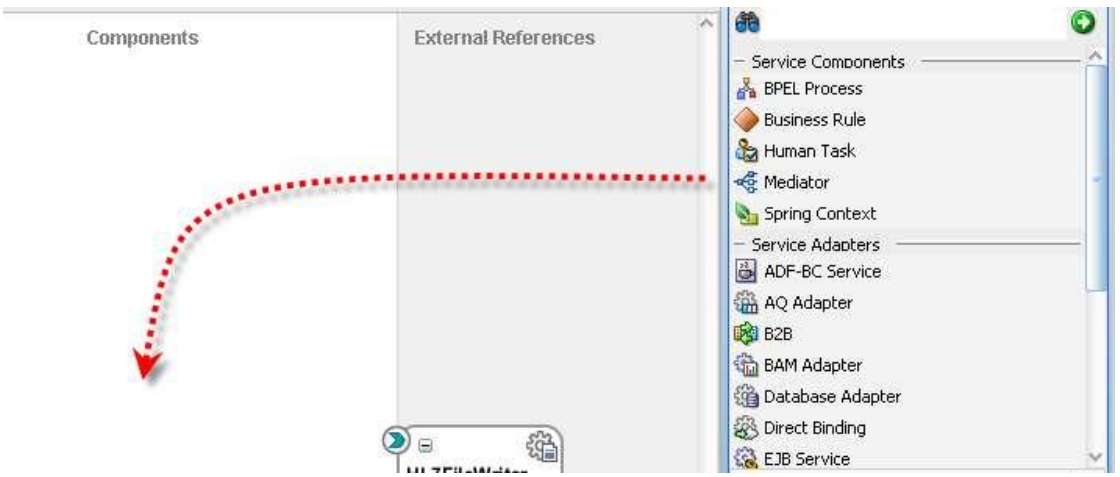
The structure of the HL7 message which will be written by the File Adapter will conform to this schema element.

Our composite has an inbound (B2B) and an outbound (File) adapter.



Now we need a piece of logic to copy the message from one adapter to the other, potentially transforming it if needed. We will not need to do this in this example since all we care about is to receive a message and write it to a file.

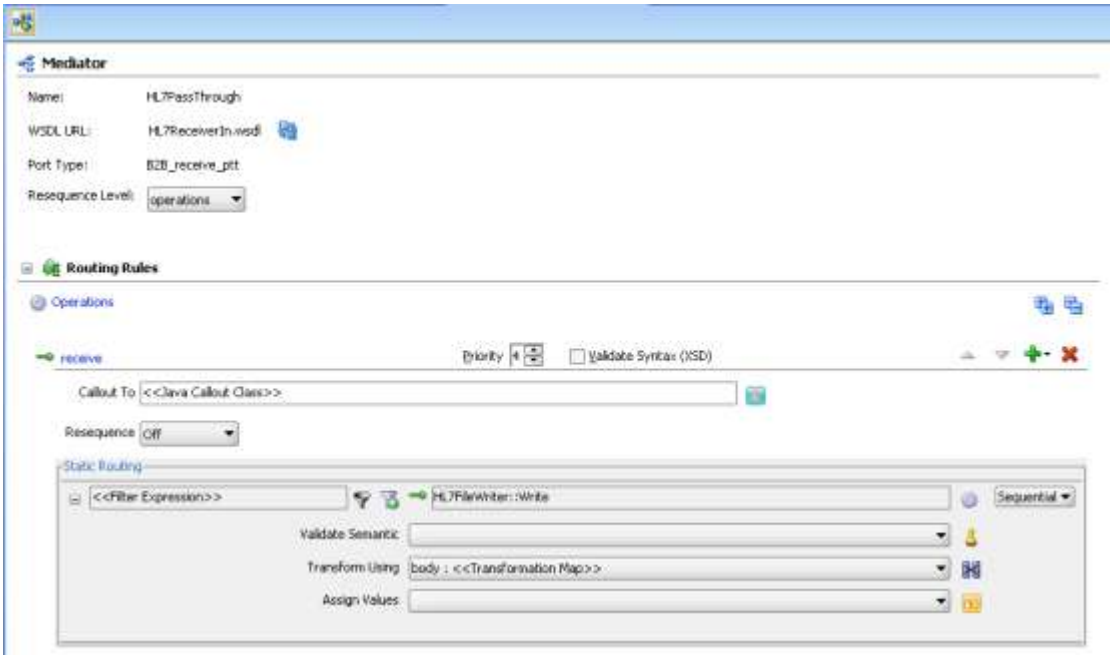
Let's choose the Mediator to develop the simple logic we need. Drag the Mediator component from the Service Components palette onto the Components swim line.



Name the component HL7PassThrough and click OK. The Mediator component appears is the composite.

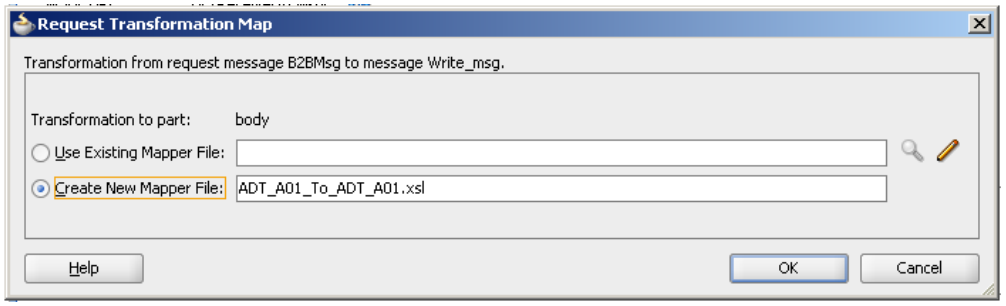
"Wire" components together by dragging from the chevron icon of the component at the left to the "corresponding" chevron icon at the component to the right.

Double-click the Mediator component to open its plan and configure it.

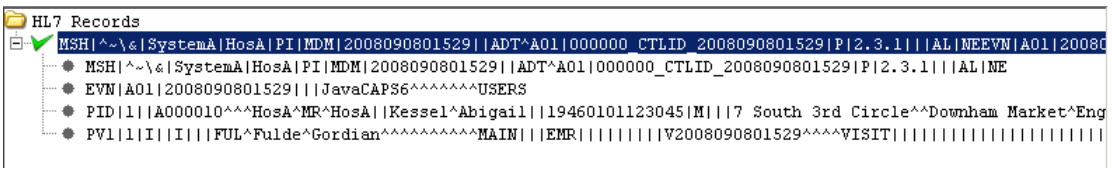


Click the Mapper icon to start defining the mapping between the B2B Adapter and the File Adapter.

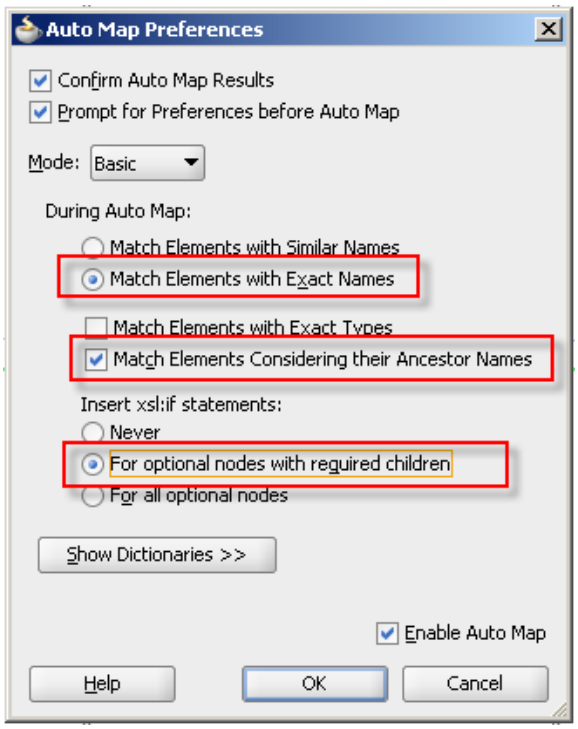
Choose "Create New Mapper File" and click OK.

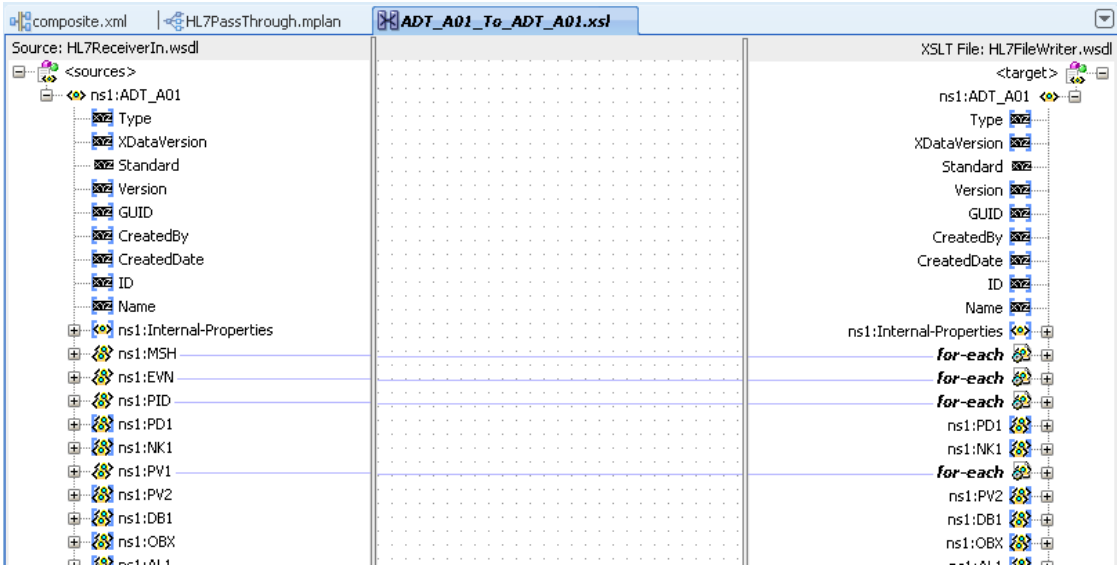


Based on your knowledge of the incoming message, map segments and fields as appropriate. In the case of our sample message we have only MSH, ENV, PID and PV1 segments.



Map MSH to MSH, EVN to EVN, PID to PID and PV1 to PV1. Ensure the confirmation dialogue box options are correctly set.



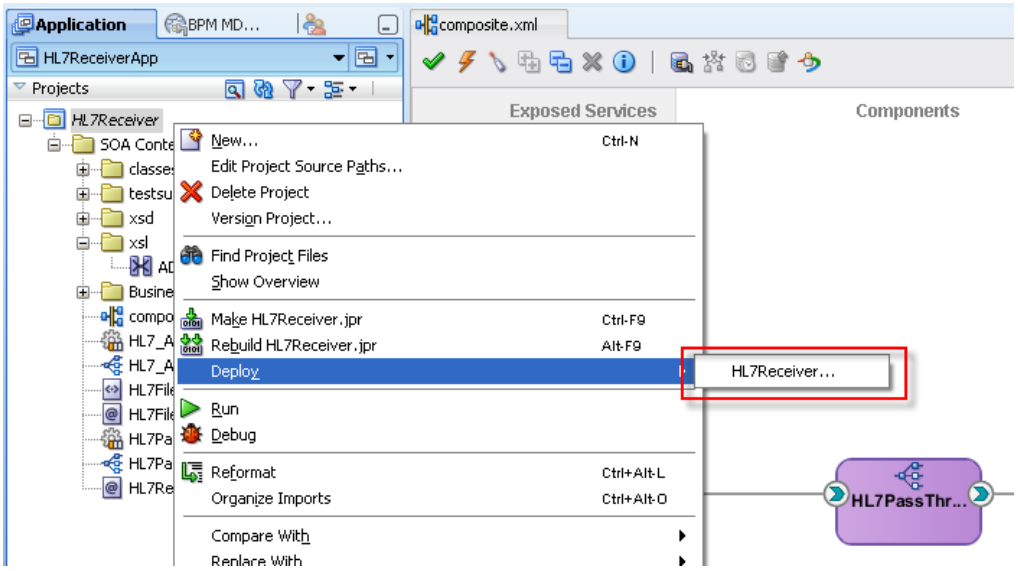


Click the Save All button in the JDeveloper tool bar and close the Mediator Plan and Mapper Tabs.

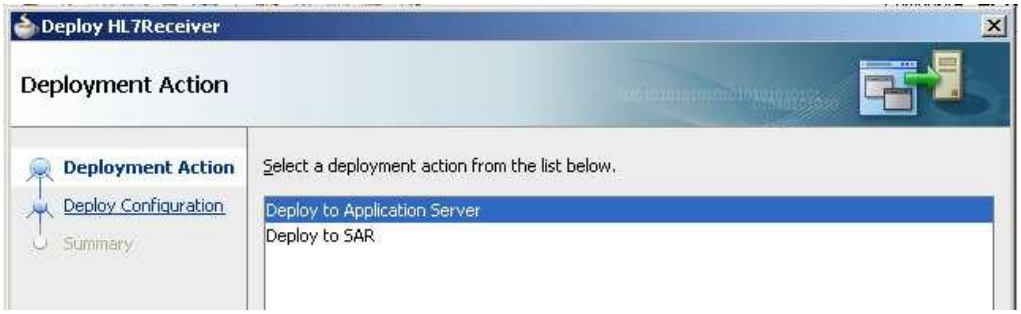
We are ready to build and deploy this application.

Make sure the Application Navigator panel is visible.

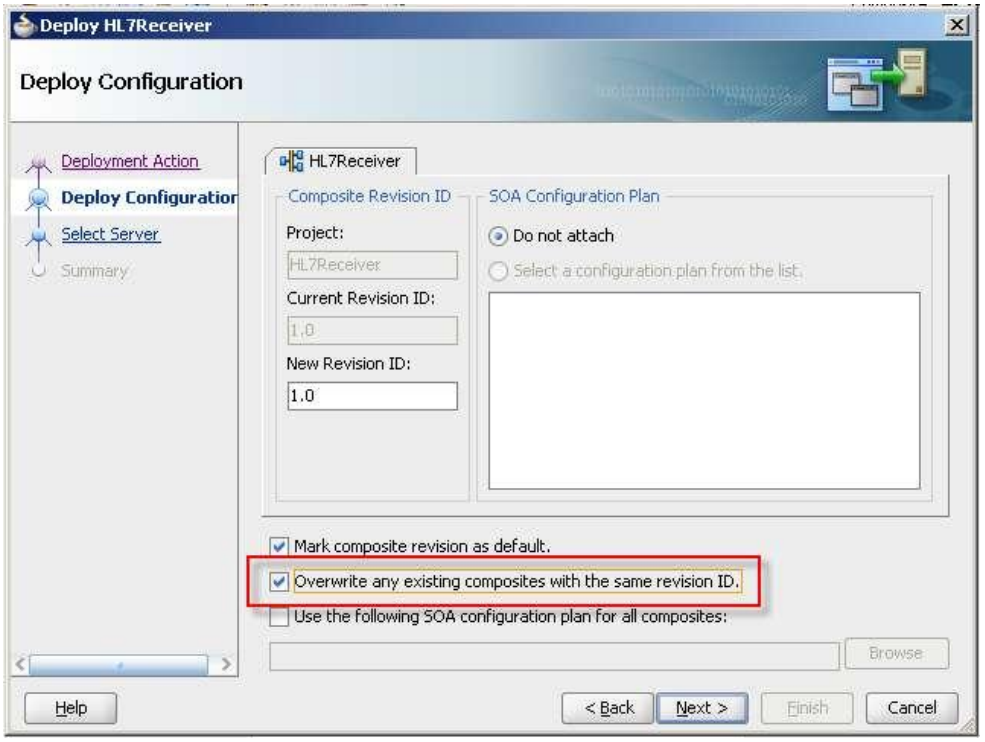
Right-click on the name of the project HL7Receiver, choose Deploy and HL7Receiver…



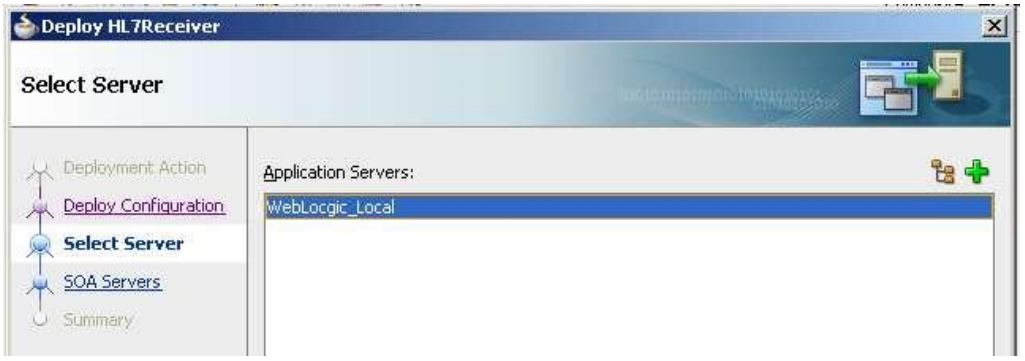
Accept default and click Next.



Check the "Overwrite any existing composites with the same revision ID" and click Next.



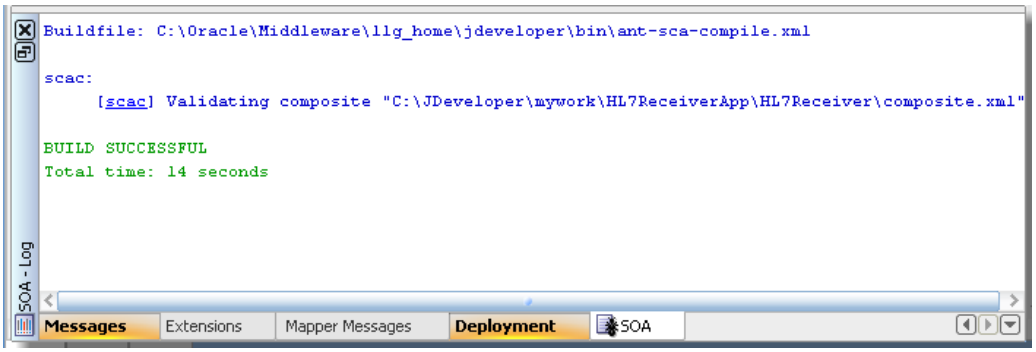
Accept the default WebLogic_Local application server and click Next.



Accept the default SOA Server and click Next.

Click Finish.

Observer SOA Log window.

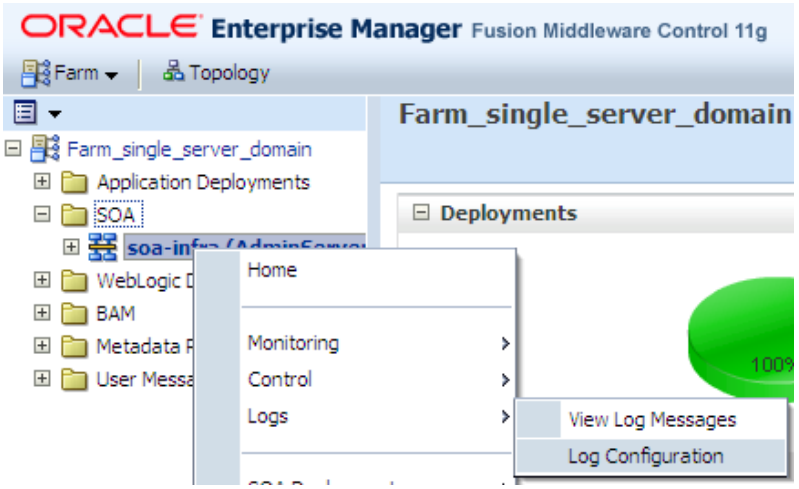


Observe Deployment Log window.

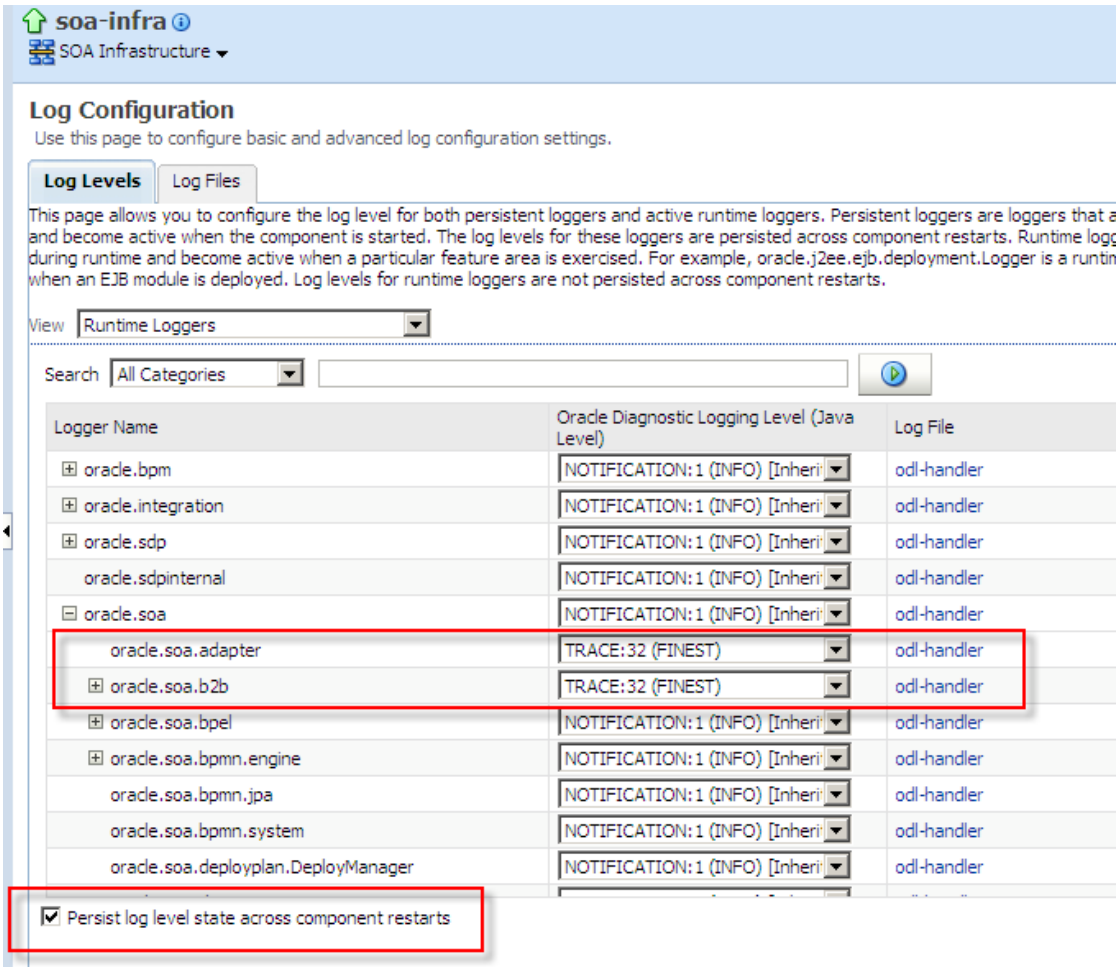
Click Save All toolbar button and exit JDeveloper Studio.

Exercise HL7 Inbound solution

To see the interaction and message exchange we need to increase logging level for the appropriate parts of the infrastructure. Start the Oracle Enterprise Manager, http://localhost:7001/em. Expand SOA node. Right-click on soa-infra. Choose Logs → Log Configuration.



Change log levels for oracle.soa.adapter and oracle.soa.b2b to FINEST, check "Persist log level state …" and click Apply.



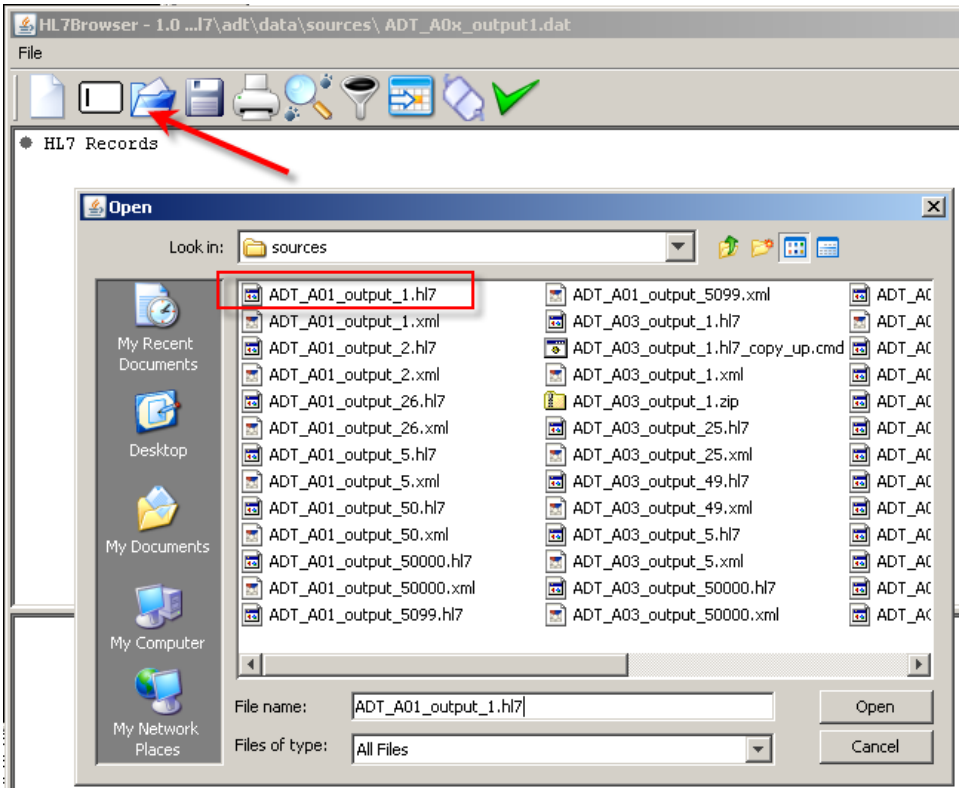
Close Enterprise Manager.

Start the B2B Trading Partner Manager Web Console, http://localhost:7001/b2b, log in and click the Reports link. Note that there are no messages shown. No messages have been received yet.

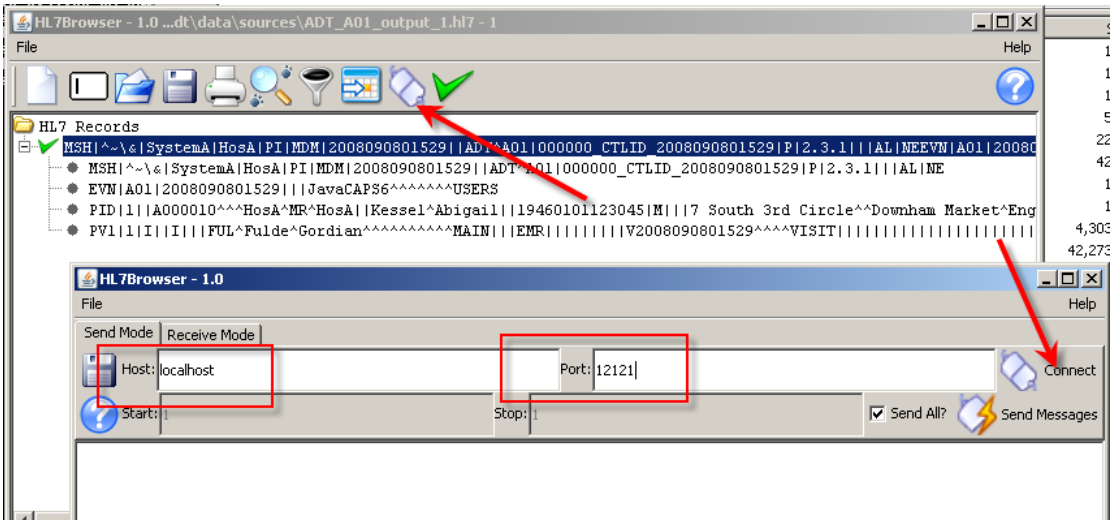
Open a command window and run the HL7Browser:

```
C:\jdk1.6.0_20\bin\java.exe -jar C:\tools\Hl7Browser.1.0\HL7Browser.jar
```

When the UI appears click the “Open an HL7 File” button, locate the ADT A01 transaction file, C:\hl7\adt\data\sources\ ADT_A0x_output1.hl7, and open it.

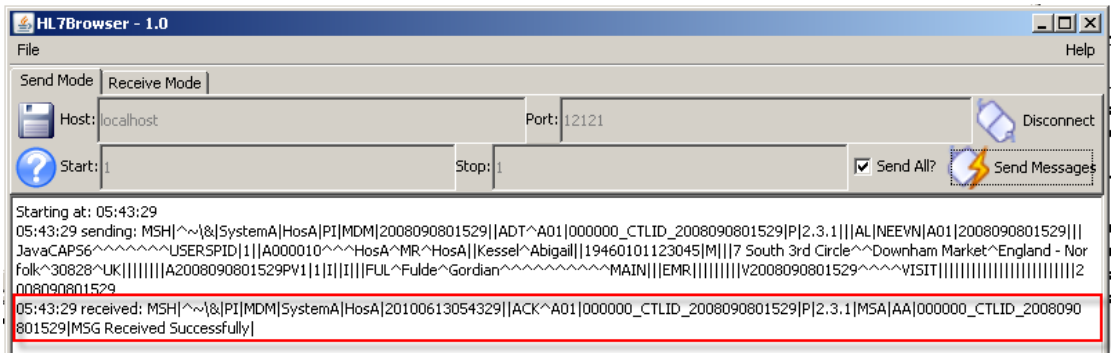


Click “Run the network utility” button, provide localhost as the Host and 12121 as the Port (recall this is the port on which the B2B listener is listening) and click Connect.

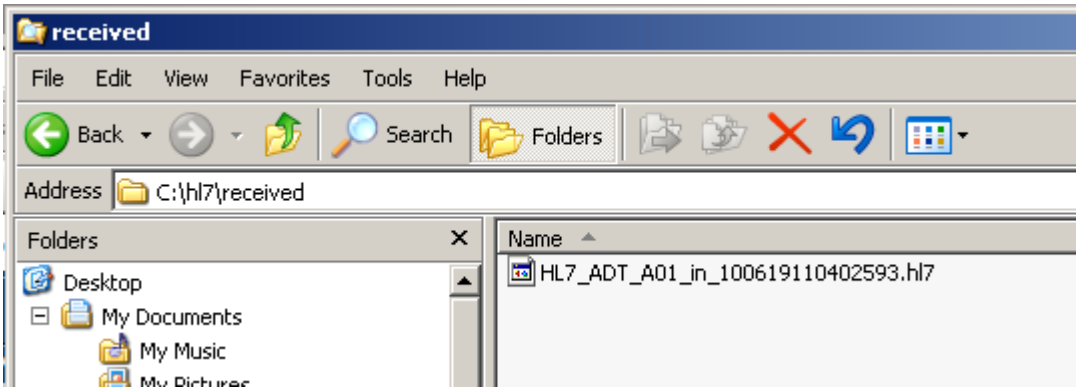


Once connected, click the “Send Messages” button.

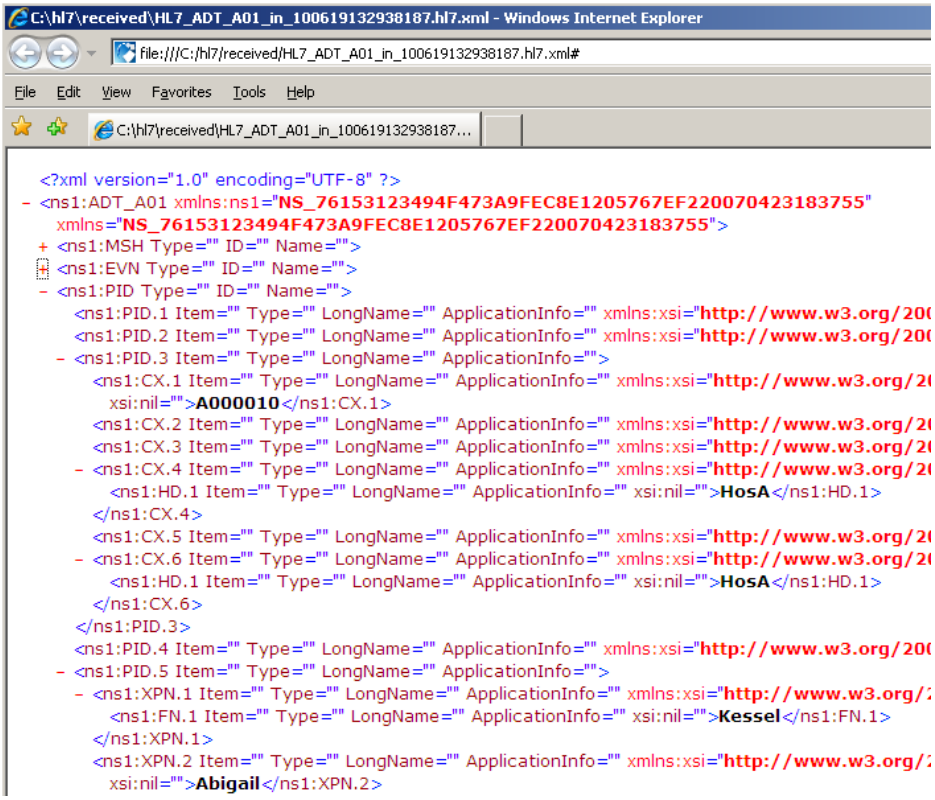
Observe message exchange. The message sent and the ACK received.



Open Windows Explorer on c:\hl7\received folder and inspect the file deposited there by the SOA Composite.

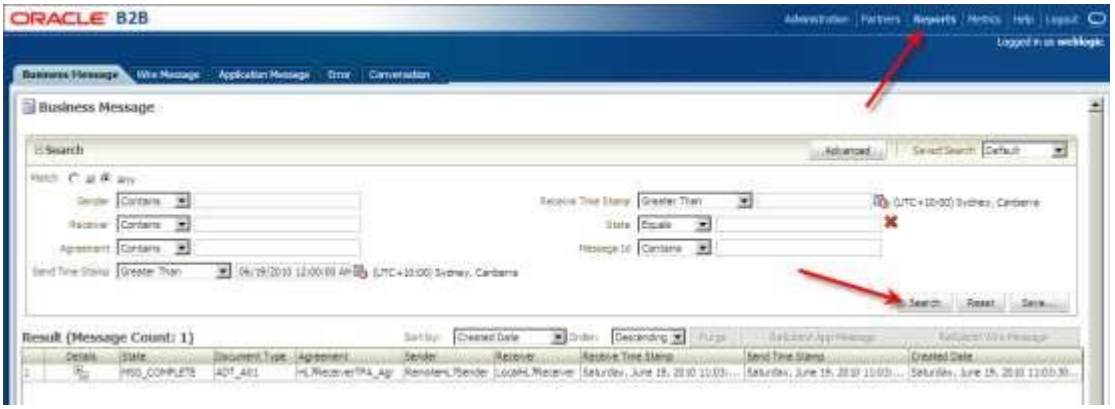


This is a XML message so use the Internet Explorer or an XML editor to inspect its contents.



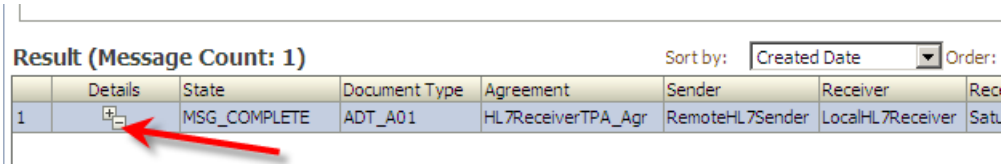
Explore Message Tracking

Start the B2B Trading Partner Manager Web Console, http://localhost:7001:b2b, and click Search.

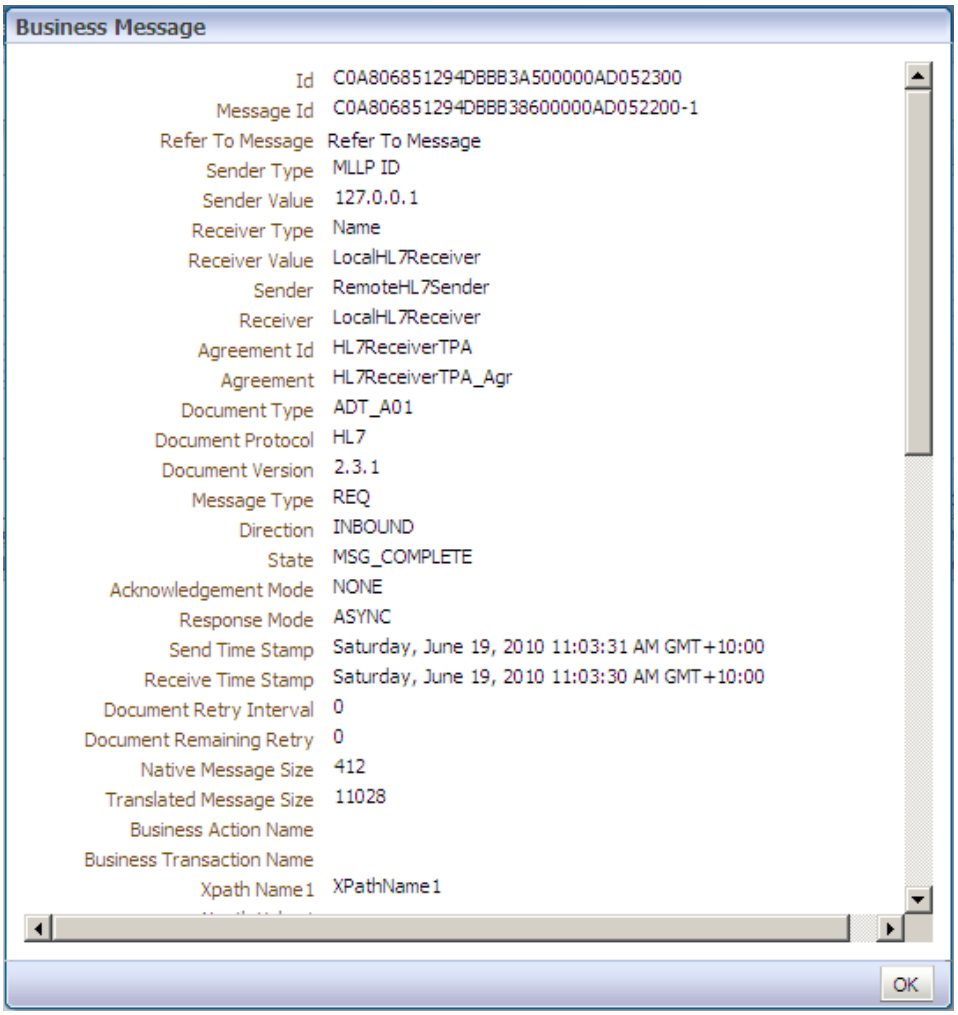


Note the status → MSG_COMPLETE, Document Type, Sender, Receiver and other details.

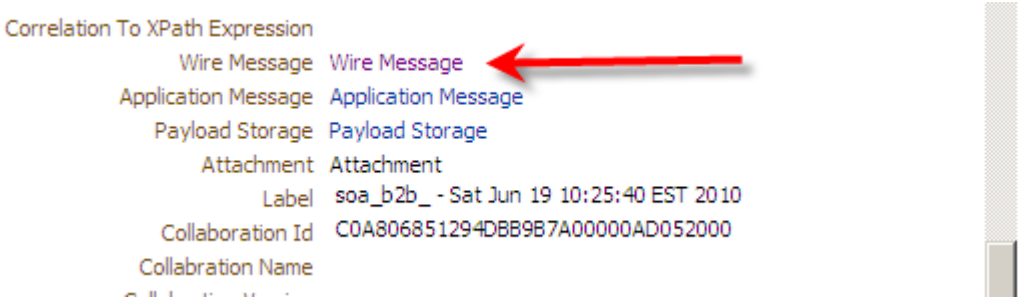
Click the Details icon.

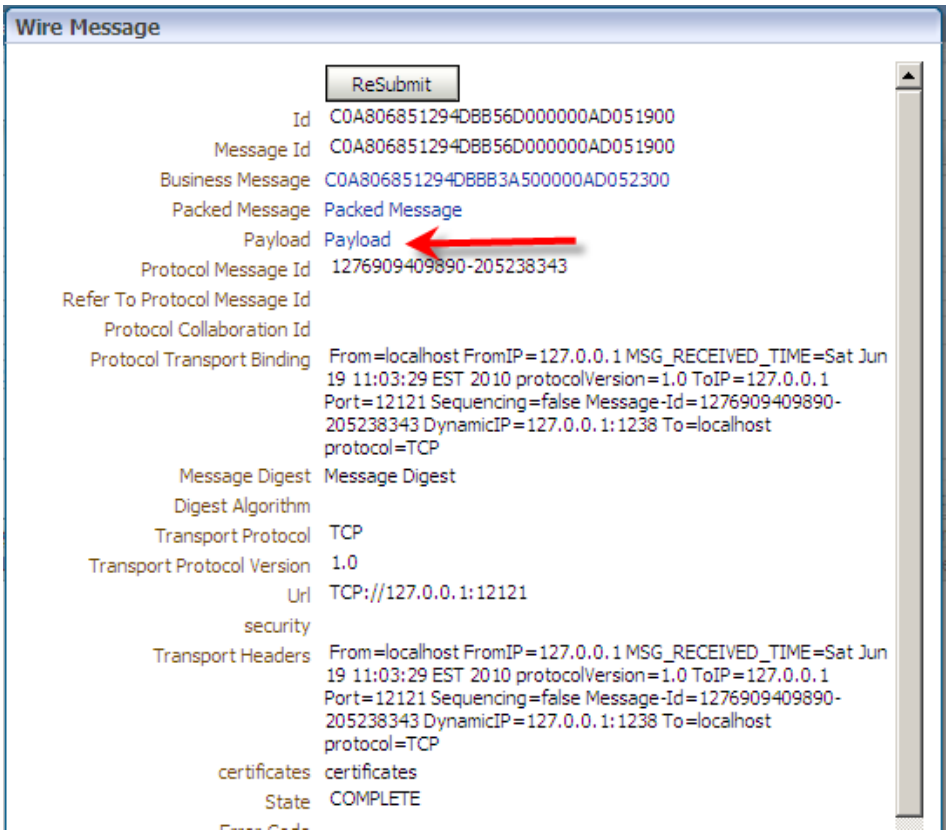


Inspect details of the "Business Message" exchange.

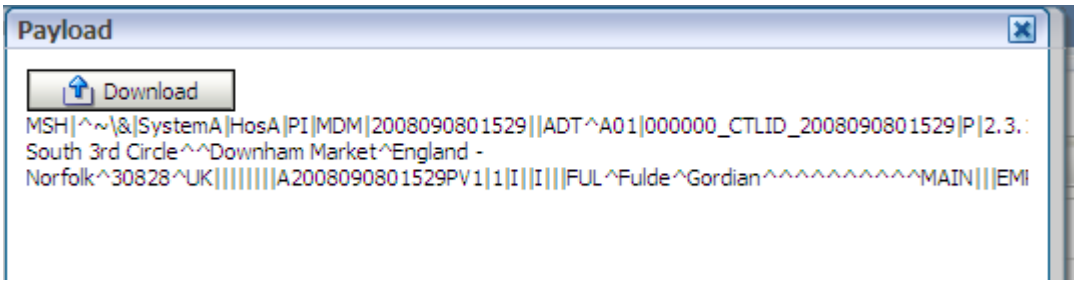


Scroll down until you see the Wire Message link. Click on it to inspect details of the message that was received over the wire.



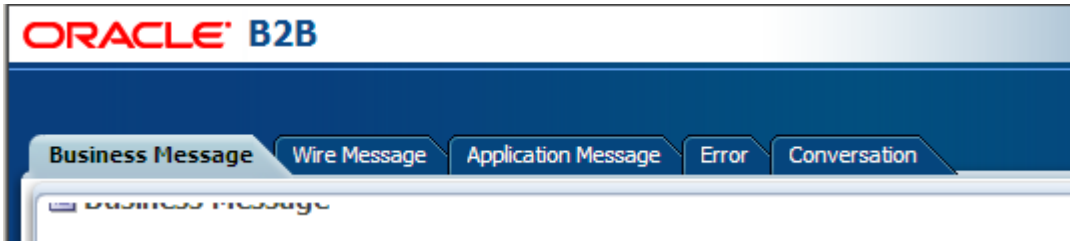


Note the ReSubmit button. You can use it to re-submit the message to the B2B infrastructure without having to have the original external system resubmit it. Click the Payload link to see the actual wire message.



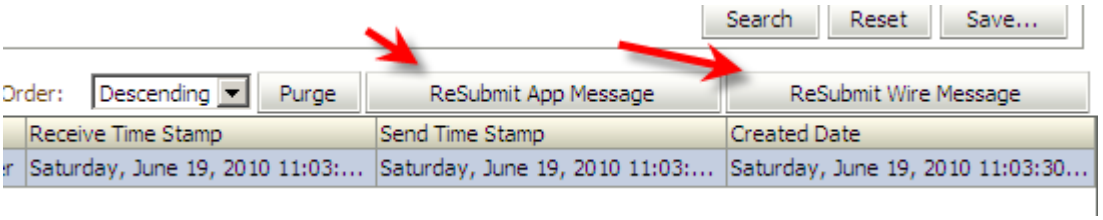
Dismiss the dialogue boxes by clicking OK on each.

Note the various sub-Tabs in the Reporting section – Business Message, Wire Message, …



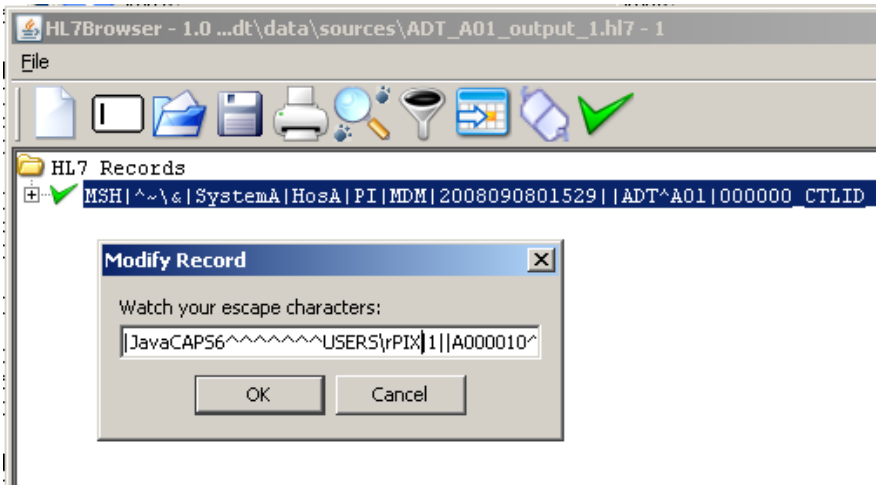
Explore the functionality to get a feel for what is available. In B2B terms this functionality is called Message Tracking.

Note, too, the facilitiy to re-submit the wire message and the application message directly from the list of messages.



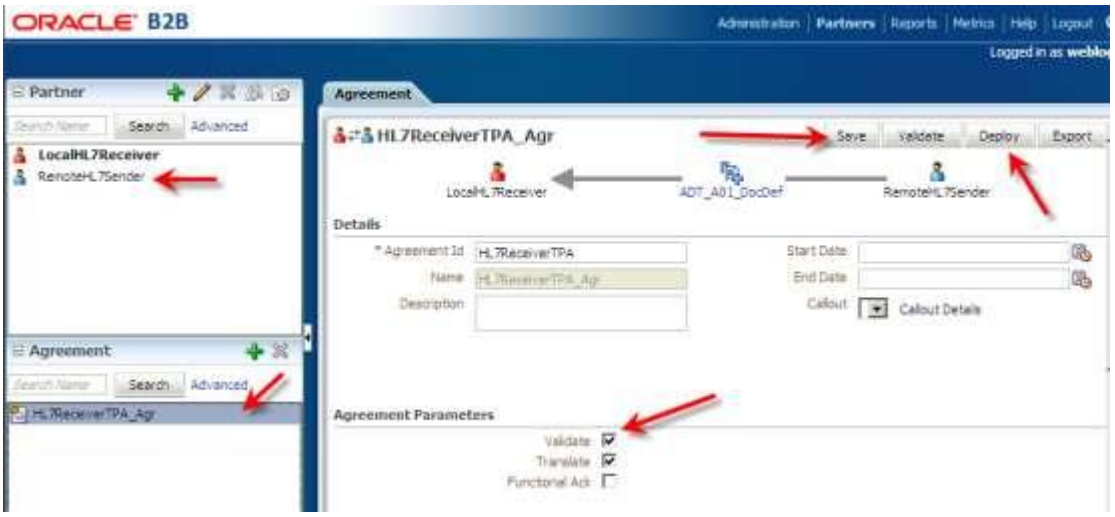
In this example the wire message is the message that is received by the B2B infrastructure and the Application Message is the message that the B2B infrastructure passes to the SOA Suite Composite for further processing.

Modify the original message in the HL7 Browser changing the PID segment ID to PIX thus making the message structurally invalid.

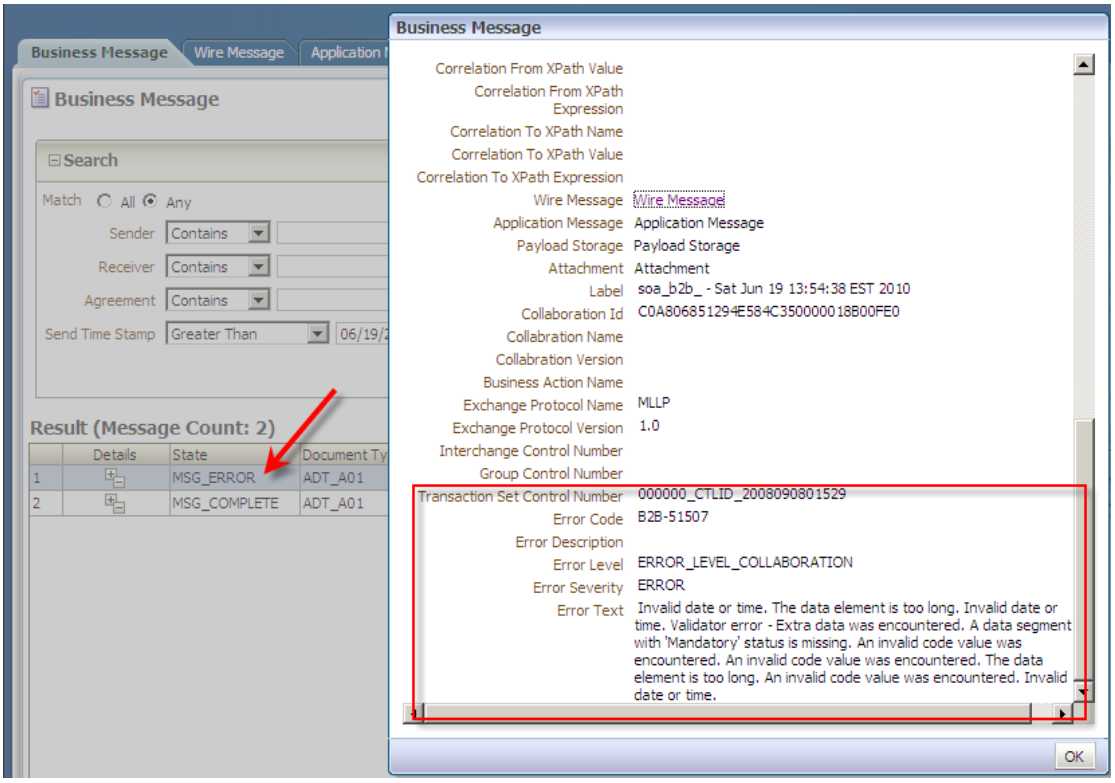


Switch to the HL7 Browser's Send Mode window and click Send Message to submit the modified message.

Switch to the B2B Console and use the Reports Tab to view the business message. Note that the status is MSG_COMPLETE even though the message is structurally invalid. This is because we did not check the "Validate" checkbox in the trading partnership agreement. Let's do that now, Save the agreement and Deploy it.

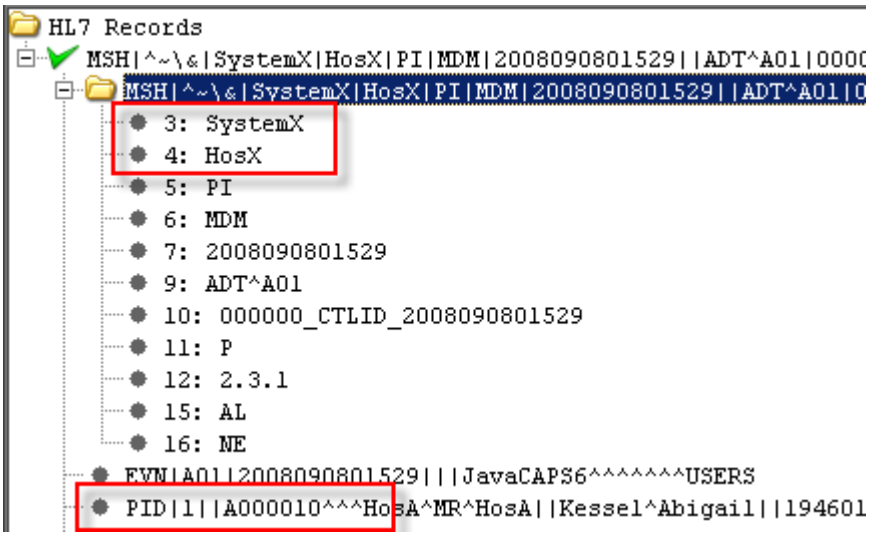


With that done, let's Send Message again and check the Business Message report in the B2B Console. Message status shows as MSG_ERROR. Clicking Details icon opens the details window with the error details.

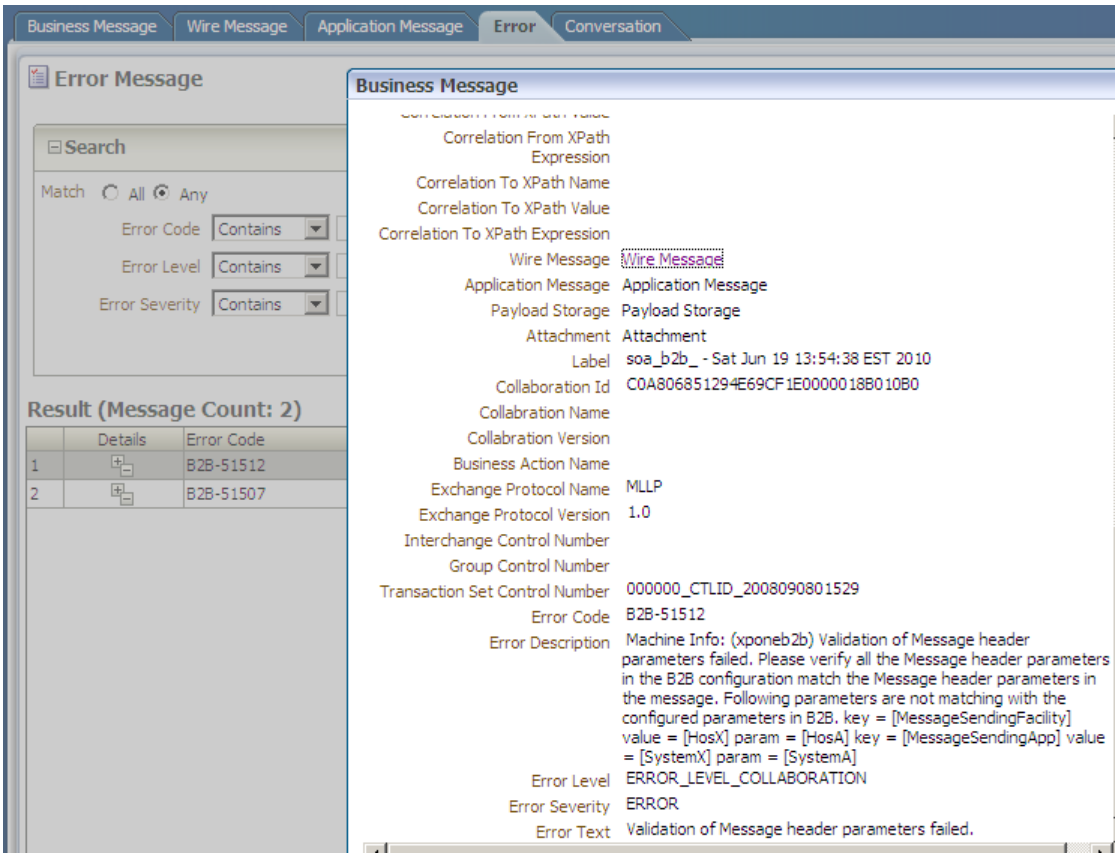


Close the Details window.

Modify the message in HL7 Browser changing PIX segment name back to PID segment name and changing MSH-3 from SystemA to SystemX and MSH-4 from HosA to HosX. Submit the message and observe the outcome in the B2B Console Report Tab.



Note that the message is in error and is shown as such in the Reports -> Error Tab with specific diagnostics.

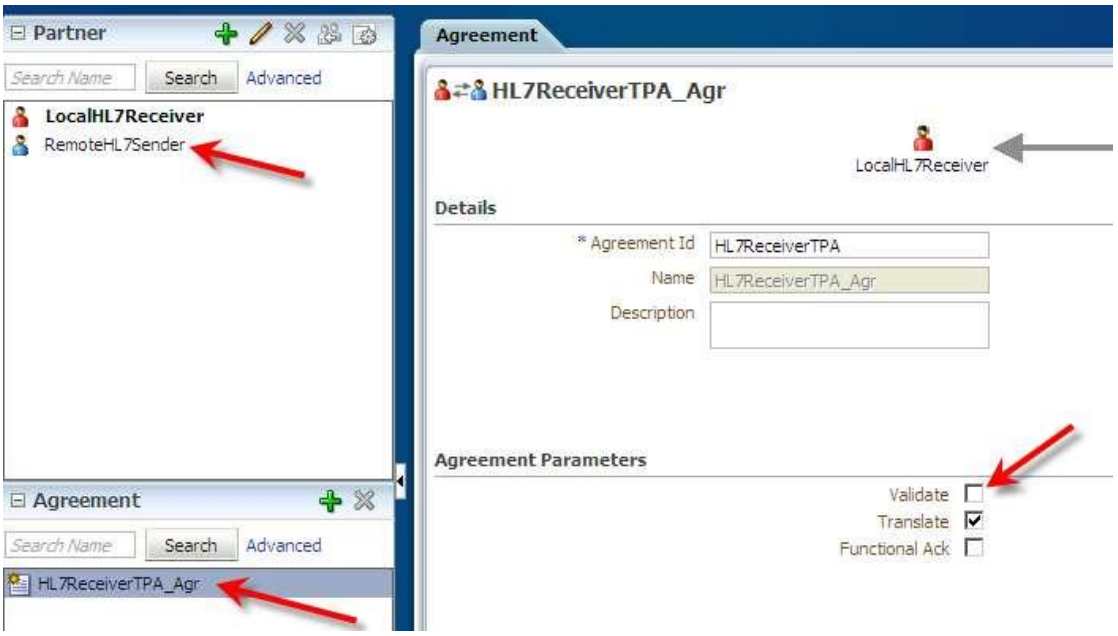


Explore Messaging Metrics

Let's submit 50 HL7 v2 messages from a file ADT_A01_output_50.hl7.

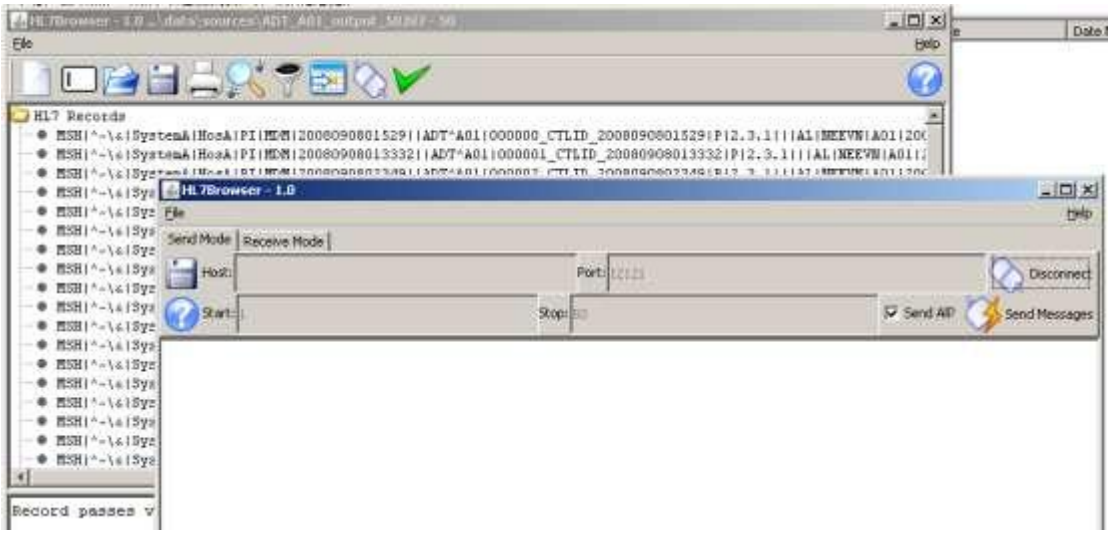
Close the HL7 Browser's sender client window and the HL7 Browser window as well to start afresh.

Switch to B2B Console and uncheck the Validate agreement parameter, save and deploy the agreement.



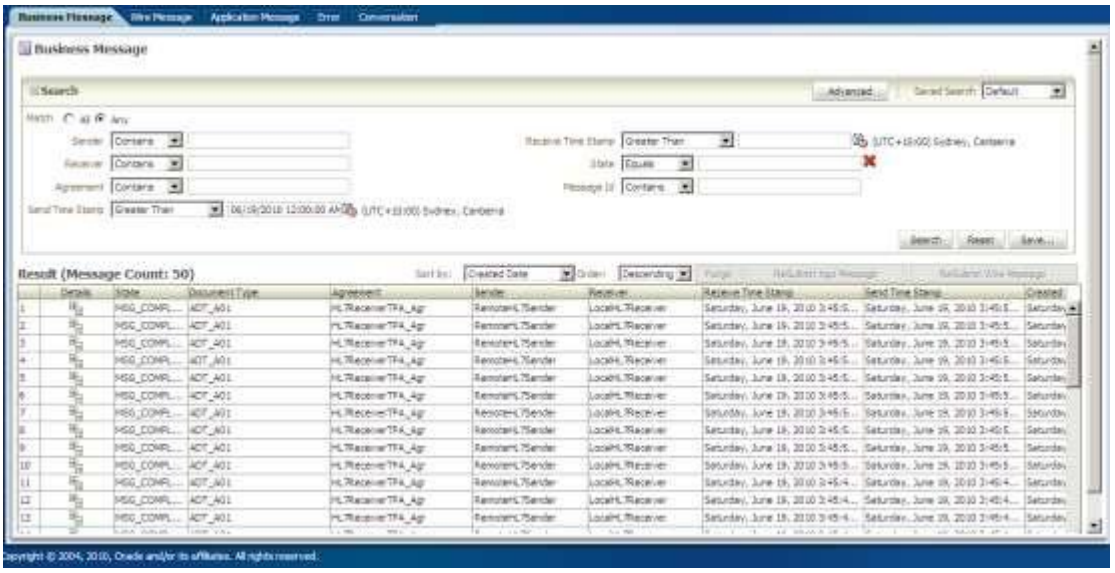
This is done so that the 50 message file, which contains transaction with data exceeding HL7 recommended lengths, does not invalidate message exchange.

Start the HL7 Browser. Open the file c:\hl7\adt\data\sources\ADT_A01_output_50.hl7.

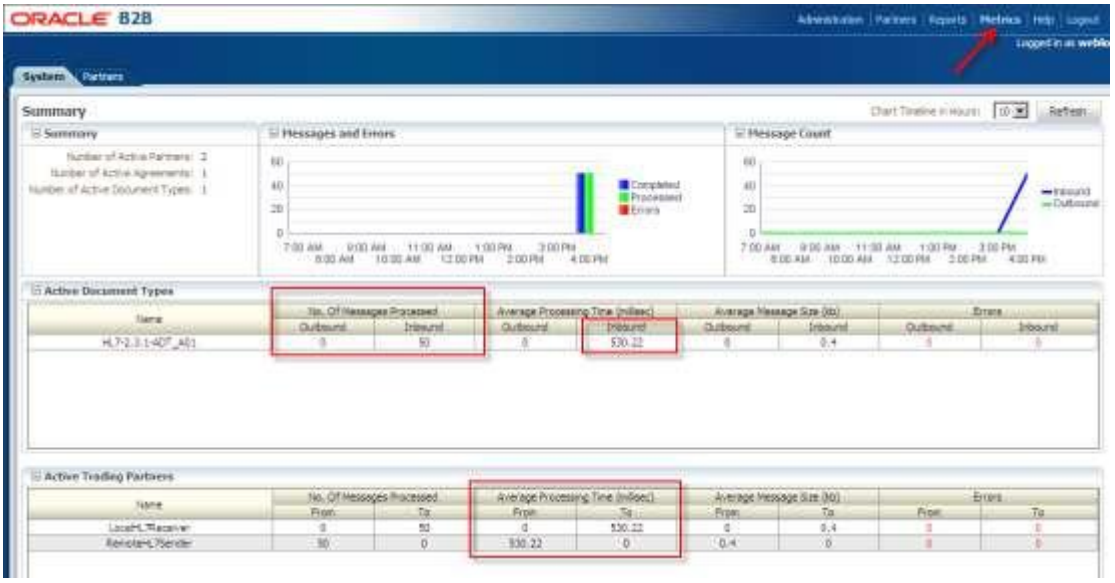


Connect and Send Messages. Observe responses in the HL7 Browser window.

Switch to B2B Console and view Reports → Business Messages.



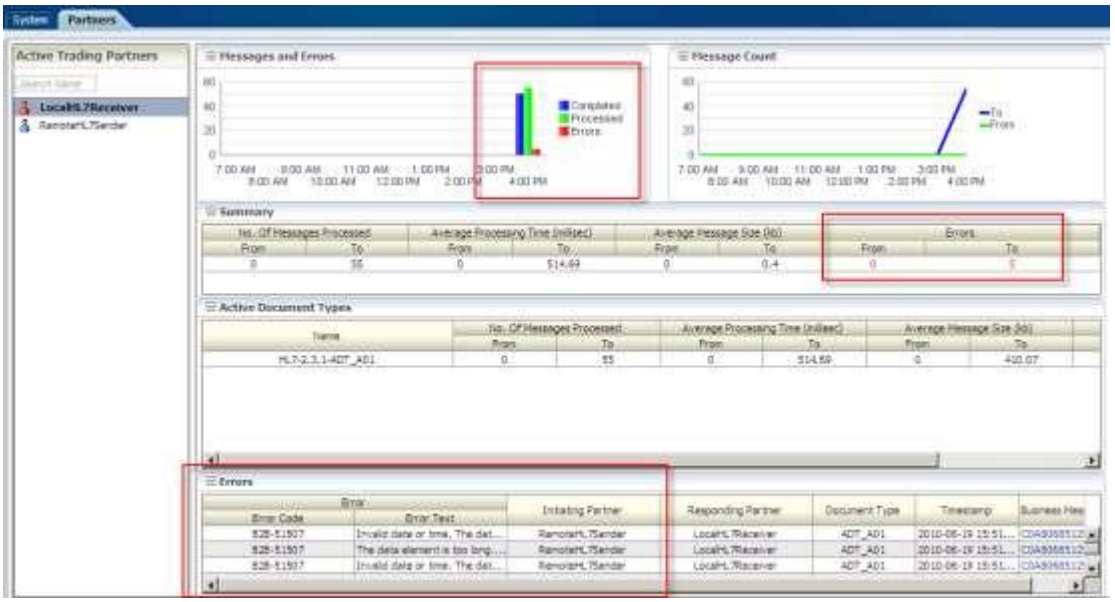
Click the Metrics link and observe the display.



Explore the information provided.

Enable Validation for the agreement, save and deploy the agreement and submit 5 messages from the file ADT_A01_output_5.hl7.

Inspect metrics for trading partner RemoteHL7Sender.



Inspecting the Server Log

The AdminServer-diagnostic.log, in the installation discussed in this article, lives in C:\Oracle\Middleware\11g_home\user_projects\domains\single_server_domain\servers\AdminServer\logs. Open the log with a text editor and look for the expression "Payload = <Exception". Inspect this message and surrounding messages. The messages we are submitting are not as valid as we might like them to be. The B2B infrastructure validates messages when asked (Validate checkbox in the trading partner agreement) and logs the diagnostics when asked. The log message might look like:

```
[2010-06-19T15:51:31.781+10:00] [AdminServer] [TRACE] [] [oracle.soa.b2b.engine] [tid:
weblogic.work.j2ee.J2EEWorkManager$WorkWithListener@2f6984] [userId: <anonymous>]
[ecid: 0000I_EkkXTCSsWFLzuHOA1C73CF0001p2,0] [SRC_CLASS:
oracle.tip.b2b.system.DiagnosticService] [APP: soa-infra] [dcid:
e6d3d61dea16942c:65bbb7b1:1294e297e08:-7fd3-0000000000000050] [SRC_METHOD:
synchedLog_J] Notification: notifyApp: payload = <Exception
xmlns="http://integration.oracle.com/B2B/Exception"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">[[
  <correlationId>null</correlationId>
  <b2bMessageId>C0A806851294EC309C50000018B07880-1</b2bMessageId>
  <errorCode>B2B-51507</errorCode>
  <errorText>
  <![CDATA[Invalid date or time.
The data element is too long.
Invalid date or time.
The data element is too long.
An invalid code value was encountered.
An invalid code value was encountered.
The data element is too long.
An invalid code value was encountered.
Invalid date or time.

  ]]>
  </errorText>
  <errorDescription>
  <![CDATA[Machine Info: (xponeb2b)
Component MSH-7.1 (time of an event) has a data type of 'string data' (ST). The
expected format was YYYY[MM[DD[HHMM[SS[.S[S[S[S]]]]]]]][+/-ZZZZ]. Segment MSH is
defined in the guideline at position 001.{br}{br}This error was detected
at:{br}{tab}Segment Count: 1{br}{tab}Field Count: 7{br}{tab}Component Count:
1{br}{tab}Characters: 29 through 42
The length of Field MSH10 (Message Control ID) is '26'. The maximum allowed length is
'20'. Segment MSH is defined in the guideline at position 001.{br}{br}This error was
```

```
detected at:{br}{tab}Segment Count: 1{br}{tab}Field Count: 10{br}{tab}Characters: 52
through 78
Component EVN-2.1 (time of an event) has a data type of 'string data' (ST). The
expected format was YYYY[MM[DD[HHMM[SS[.S[S[S[S]]]]]]]][+/-ZZZZ]. Segment EVN is
defined in the guideline at position 002.{br}{br}This error was detected
at:{br}{tab}Segment Count: 2{br}{tab}Field Count: 2{br}{tab}Component Count:
1{br}{tab}Characters: 103 through 116
The length of Field PID3 (Patient Identifier List) is '22'. The maximum allowed length
is '20'. Segment PID is defined in the guideline at position 003.{br}{br}This error
was detected at:{br}{tab}Segment Count: 3{br}{tab}Field Count: 3{br}{tab}Characters:
148 through 170
Field PV14 (Admission Type) does not contain a valid identification code: 'I' is not
allowed. Segment PV1 is defined in the guideline at position 006.{br}{br}This error
was detected at:{br}{tab}Segment Count: 4{br}{tab}Field Count: 4{br}{tab}Characters:
300 through 301
Component PV17-14 (identifier type code) does not contain a valid identification code:
'MAIN' is not allowed. Segment PV1 is defined in the guideline at position
006.{br}{br}This error was detected at:{br}{tab}Segment Count: 4{br}{tab}Field Count:
7{br}{tab}Component Count: 13{br}{tab}Characters: 331 through 335
The length of Field PV119 (Visit Number) is '23'. The maximum allowed length is '20'.
Segment PV1 is defined in the guideline at position 006.{br}{br}This error was
detected at:{br}{tab}Segment Count: 4{br}{tab}Field Count: 19{br}{tab}Characters: 350
through 373
Component PV119-5 (identifier type code) does not contain a valid identification code:
'VISIT' is not allowed. Segment PV1 is defined in the guideline at position
006.{br}{br}This error was detected at:{br}{tab}Segment Count: 4{br}{tab}Field Count:
19{br}{tab}Component Count: 5{br}{tab}Characters: 368 through 373
Component PV1-44.1 (time of an event) has a data type of 'string data' (ST). The
expected format was YYYY[MM[DD[HHMM[SS[.S[S[S[S]]]]]]]][+/-ZZZZ]. Segment PV1 is
defined in the guideline at position 006.{br}{br}This error was detected
at:{br}{tab}Segment Count: 4{br}{tab}Field Count: 44{br}{tab}Component Count:
1{br}{tab}Characters: 398 through 411

  ]]>
  </errorDescription>
  <errorSeverity>2</errorSeverity>
  <errorDetails>
<parameter name="b2b.messageId" value="C0A806851294EC309C50000018B07880-1"/>
<parameter name="b2b.documentTypeName" value="ADT_A01"/>
<parameter name="b2b.documentProtocolVersion" value="2.3.1"/>
<parameter name="b2b.documentDefinitionName" value="ADT_A01_DocDef"/>
<parameter name="b2b.documentProtocolName" value="HL7"/>
<parameter name="b2b.messageType" value="1"/>
<parameter name="b2b.fromTradingPartnerId" value="127.0.0.1"/>
<parameter name="b2b.fromTradingPartnerIdType" value="MLLP ID"/>
<parameter name="b2b.toTradingPartnerId" value="LocalHL7Receiver"/>
<parameter name="b2b.toTradingPartnerIdType" value="Name"/>
  </errorDetails>
</Exception>
]]
```

This concludes the exercise.

Summary

Oracle SOA Suite B2B component can be used to provide HL7 v2 messaging support for healthcare environments.

In this article a simple Oracle SOA Suite 11g B2B infrastructure-based HL7 v2 Receiver project for ADT A01 messages was developed and exercised. Message Tracker was used to view messages, message states and messaging performance.

References

[1] Oracle B2B Site, Available: http://www.oracle.com/technology/products/integration/b2b/index.html, Accessed: 4 June 2010

[2] Oracle B2B User's Guide, Available: http://download.oracle.com/docs/cd/E15523_01/integration.1111/e10229.pdf, Accessed: 19 June 2010