

Oracle SOA Suite 11g R1 PS3 HL7 Messaging

HL7 v2.3.1 A19 Query Processor

Michael Czapski
April 2011
Revision 1.0.0

Table of Contents

Introduction.....	2
Changes from PS2.....	2
Assumptions	2
Note	2
Prerequisites to download	2
Overview of the Process	3
B2B Configuration	4
<i>Create and Export "Specs"/documents</i>	4
<i>"Introduce" New Documents</i>	5
<i>Configure the "Self" Partner Profile</i>	6
<i>Create and Configure "Remote" Partner</i>	7
<i>Configure Inbound QRY Agreement</i>	9
<i>Configure Outbound ADR Agreement</i>	10
<i>Verify and Set Global B2B Configuration</i>	12
SOA Composite Configuration	12
<i>Create a B2BA19Queue</i>	12
<i>A19Processor Application</i>	15
<i>QRY Receiver Project</i>	16
<i>ADR Sender Project</i>	27
<i>Test the Solution</i>	34
<i>Summary</i>	34

Introduction

The Oracle SOA Suite 11g R1 B2B functionality can be used for HL7 v2.x delimited messaging, both inbound and outbound. I have a series of articles which provide step-by-step instructions for developing HL7 v2.x delimited messaging solutions for processing inbound and outbound messages, with varying ACK patterns -

<http://blogs.czapski.id.au/?s=hl7+soa+suite>.

This article discusses how an A19 Query processing solution can be implemented using the SOA Suite 11g R1 PS3.

We have a client sending a HL7 v2.3.1 A19 QRY request, asking for demographic details for a patient specified by an ID. The HL7 v2.3.1 A19 ADR response will carry a PID segment with basic demographics. The client identifies itself with MSH-3 (Application ID) of "A19QRY" and MSH-4 (Facility ID) of "CL11". The query processor is identified by the client with MSH-5 (Application ID) of "A19ADR" and MSH-6 (Facility ID) of "GWYQ".

Changes from PS2

PS3 Partnership Agreement configuration panel adds the "B2B Handles Functional ACK" drop-down. Selecting "No" will configure the infrastructure in such a way that it will not generate the Functional ACK on the inbound side and will expect the back-end solution to return the Functional ACK for this partnership only. In PS2 this setting was global for all partnerships. The global setting is still there but the per-partnership setting is much better when only one or few partnerships require back-end generated ACK.

Assumptions

It is assumed that you have the Oracle SOA Suite 11g R1 PS3 installed and running. There are a number of Internet-borne resources which discuss where to get the software and how to install it.

It is assumed that you have the Oracle B2B Document Editor installed. It is a separate download so just because you have the SOA Suite installed it does not mean you have the B2B Document Editor. The Oracle site will provide links to the software and documentation.

If you have the SOA Suite 11g R1 PS2 then you will need to set the global Administration-->Configuration-->B2B Handles Functional ACKs to FALSE and all the partnership will obey that. This means that you will have to have functional ACK generated by the back-end for all agreements you have or will have deployed to the one server or that you will have to have separate servers for the two different ways in which functional ACKs are to be handled.

Note

HL7 XML Schema Documents (XSDs) are very large and very complex. I found it necessary to allocate 6.5GB of memory to my 64-bit Windows XP VM, and to tune the Sun JVM memory parameters for the WebLogic server very aggressively, to allow both the WebLogic Server, the 64-bit JDeveloper and the rest of my development kit to run all at the same time. I would have loved to give the VM more memory but also there was no more to give. I suspect that one will have a very hard time trying to run all that needs to run in less than that amount.

Prerequisites to download

To simulate the A19 QRY sender I use the HL7 Sender client discussed in the blog article "[HL7 Sender, HL7 Listener and HL7 Proxy – developer tools I always wanted](#)", at

<http://blogs.czapski.id.au/2010/12/hl7-sender-hl7-listener-and-hl7-proxy-developer-tools-i-always-wanted>. Feel free to download and use the code. If you have a different client, which you prefer, use that.

For obscure technical reason one cannot have the QRY and the ADR structures referenced in the same SOA Composite project. I hope this will get resolved at some point in the not-too-distant future. In the meantime one must have two separate projects and have one pass a message, whose structure is different from either, to the other. To save myself the trouble I use a QRY definition from the HL7 v2.3.1 XSDs download at <http://wiki.open-esb.java.net/attach/HL7/hl7v2xsd.zip>. Download the archive and extract it to a convenient location. In this article I only use the QRY_A19.xsd, segments.xsd, fields.xsd and datatypes.xsd from version 2.3.1.

Download the test message file from

http://blogs.czapski.id.au/wp-content/uploads/2011/04/QRY_A19_ID_D224_3716691_HL7.2.3.1.in.zip and unzip it to a convenient location, for example C:\hl7\2.3.1_A19. The QRY message will be used to exercise the solution.

Overview of the Process

As with any SOA Suite HL7 v2.x delimited messaging solution there are specific a series of steps to follow. These steps, in overview, are enumerated below. Each step will be discussed and illustrated later.

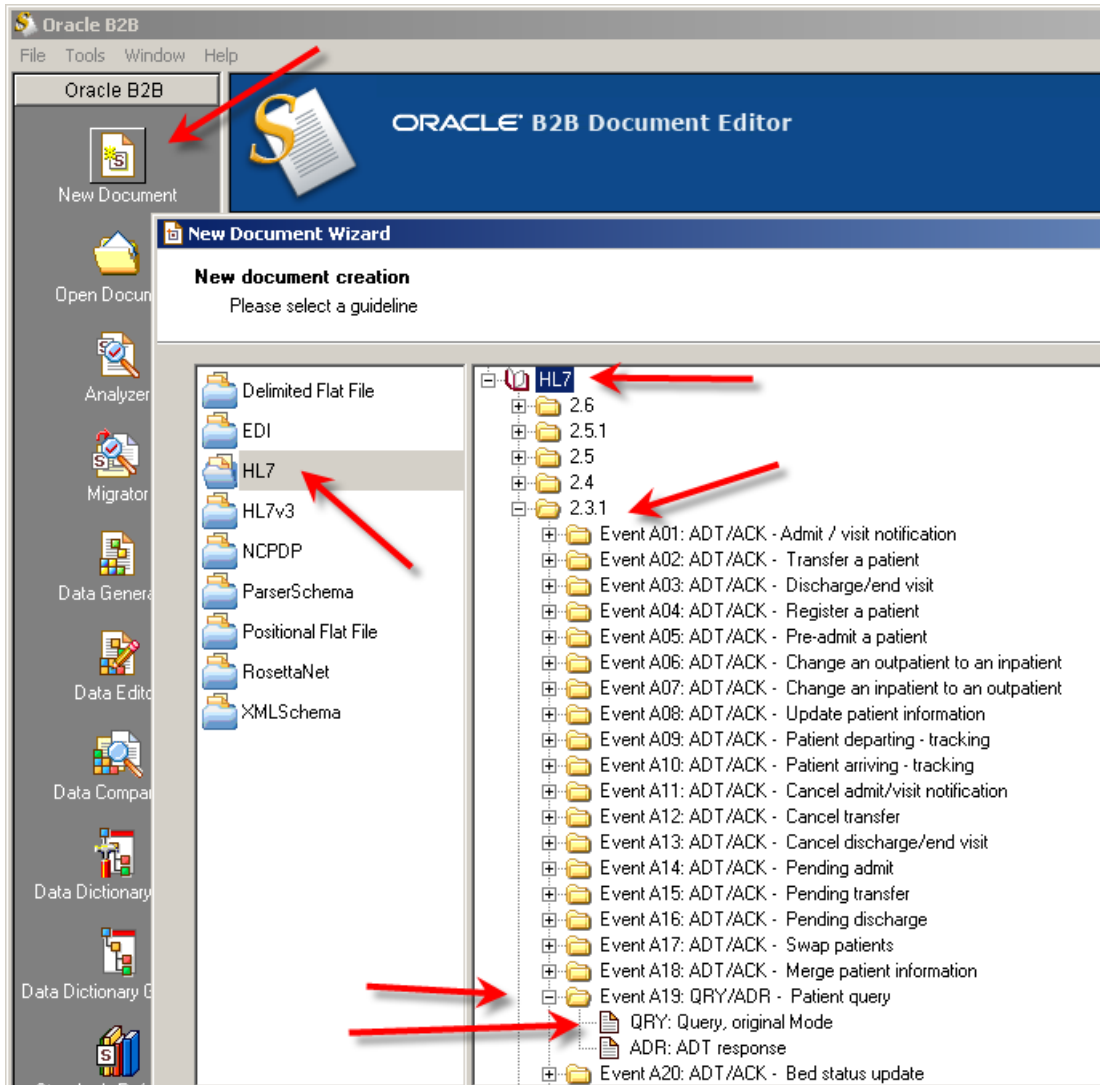
1. Create and Export "Specs" representing the HL7 message structures to be used. This is done using the B2B Document Editor. For this article the "specs"/documents/message structures will be QRY A19 and ADR A19, both v2.3.1.
2. "Introduce" new documents to the B2B infrastructure. This is trading partner-independent and is done using the B2B Web Console.
3. If needs be, configure new "HL7 Message Facility ID" and "HL7 Message Application ID" values in the "self" partner profile, to allow it to identify A19 query messages originating at specific A19 client(s). This is done using the B2B Web Console.
4. Create and configure new Partner(s), one for each distinct (MSH-3+MSH-4) A19 query client.
 - a. Configure Partner --> Profile Identifiers
 - b. Configure Partner --> Documents
 - c. Configure Partner --> Channels
5. For each distinct Partner create, configure and deploy Trading Partner Agreement for the inbound QRY message
6. For each distinct Partner create, configure and deploy Trading Partner Agreement for the inbound QRY message
7. For each distinct Partner create, configure and deploy Trading Partner Agreement for the outbound ADR message
8. Create and deploy a back-end solution to receive QRY queries and return ADR responses - for this article this will be done using the JDeveloper IDE

9. Test the solution

B2B Configuration

Create and Export "Specs"/documents

Start the B2B Document Editor, click "New Document", select HL7-->HL7-->2.3.1-->Event A19: QRY/ADR - Patient query-->QRY: Query, original Mode, click Next and Finish



Save the guideline in a convenient folder naming it "ADT_A19_QRY_231.ecs".

Pull down the File menu, choose Export..., select "Oracle B2B 2.0", click Next, Next and Finish

Click "New Document", select HL7-->HL7-->2.3.1-->Event A19: QRY/ADR - Patient query-->ADR: ADT response, click Next and Finish

Save the guideline in a convenient folder naming it "ADT_A19_ADR_231.ecs".

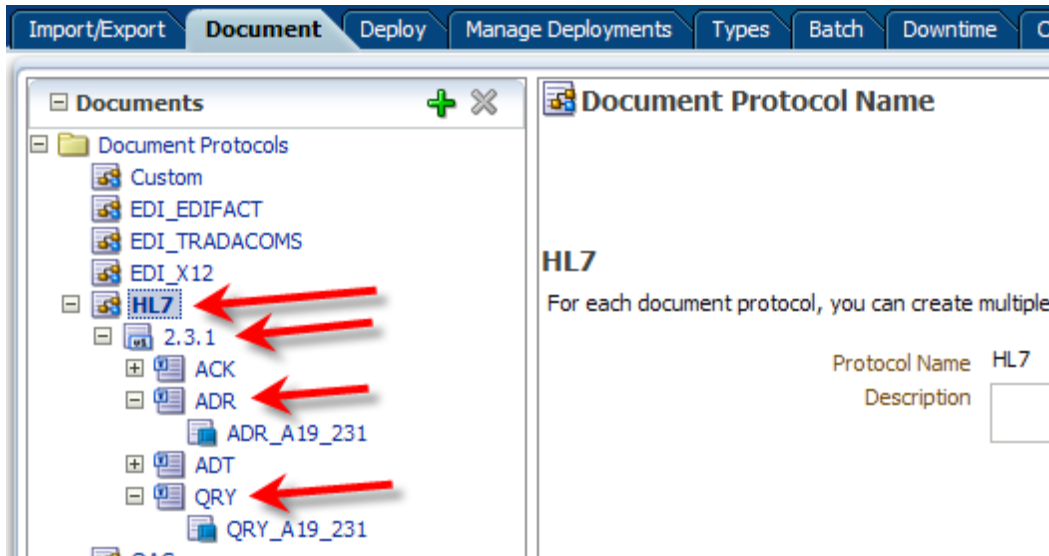
Pull down the File menu, choose Export..., select "Oracle B2B 2.0", click Next, Next and Finish

Close the B2B Document Editor - we are done with it.

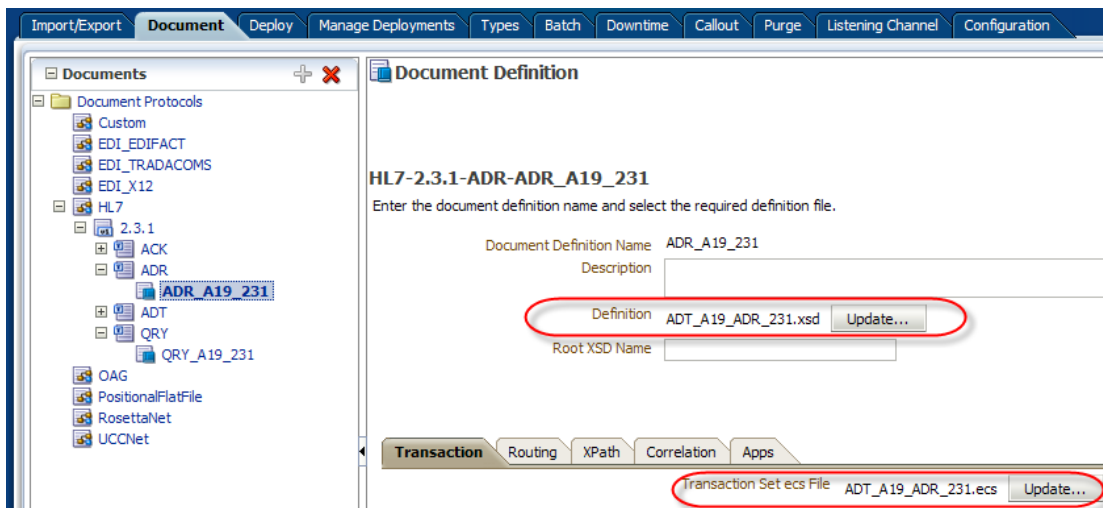
"Introduce" New Documents

Start the B2B Web Console, typically <http://localhost:7001/b2bconsole>, then select Administration-->Document

Create the document hierarchies HL7-->2.3.1-->ADR and HL7-->2.3.1-->QRY

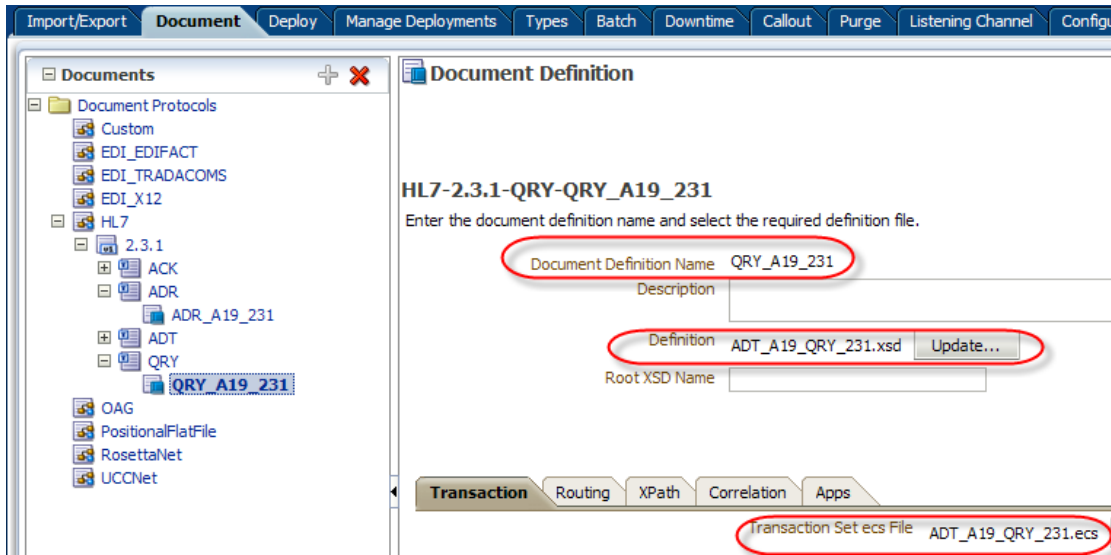


Under ADR add a new document definition, named "ADR_A19_231", using "ADT_A19_QRY_231.xsd" and "ADT_A19_QRY_231.ecs".



In the Routing Tab enter "ADR_A19_231_RID" as Routing ID. Save.

Under QRY add a new document definition, named "QRY_A19_231", using "ADT_A19_QRY_231.xsd" and "ADT_A19_QRY_231.ecs".



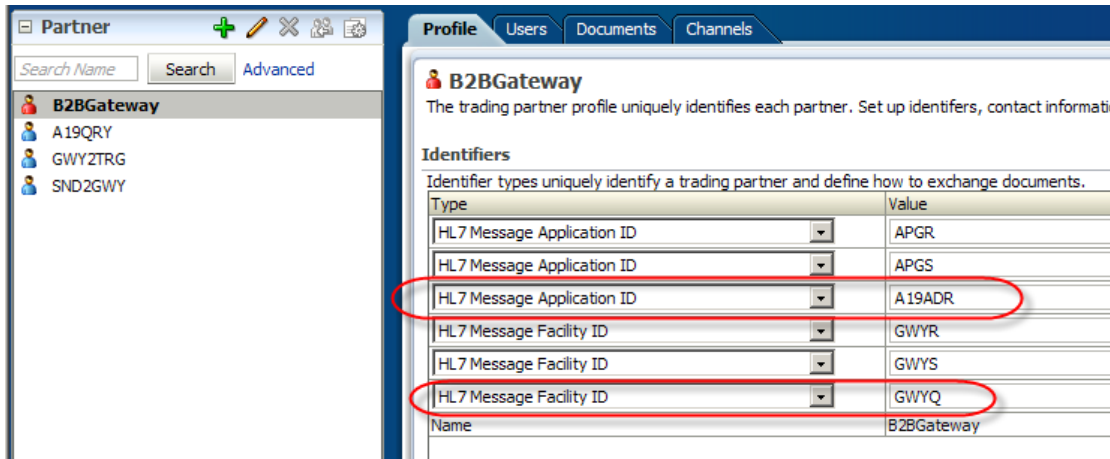
In the Routing Tab enter "QRY_A19_231_RID" as Routing ID. Save.

The B2B infrastructure now "knows" about the new document definitions.

Configure the "Self" Partner Profile

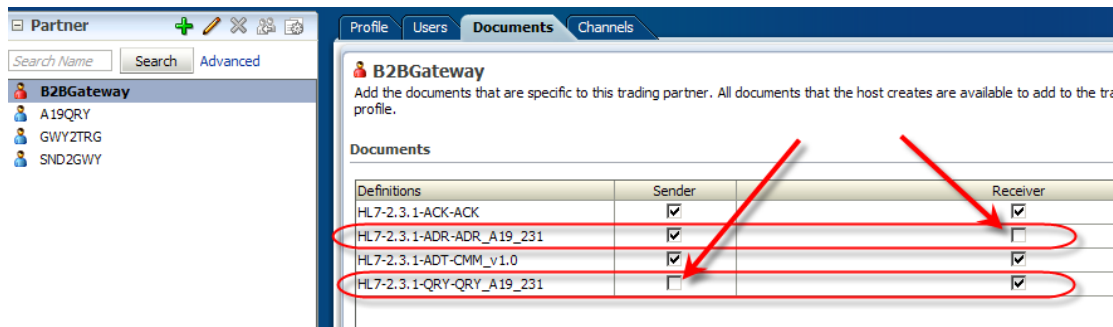
Oracle B2B has a notion of Partners, which are engaging in exchange of business documents/messages. There is always a single "local" partner, which represents the enterprise hosting the B2B infrastructure (I call this partner "self" in this article, regardless of the actual name it is given in the B2B Web Console's configuration) , and one or more "remote" partners, with whom messages are exchanged. Incoming message would be coming from a "remote" partner to the "local" partner. Outgoing message would be coming from the "local" partner to the "remote" partner. Information specific to the partner is associated with the partner, so the identifiers used to identify the "local" partner in message exchanges will be configured under the "local" partner profile and information associated with the remote partner, for example the local host and port on which the infrastructure listens for incoming messages from the specific remote partner will be configured under the remote partner's profile. A "Partnership Agreement", another concept in the B2B infrastructure, defines the parameters used for exchange of messages in one direction between the "local" and one "remote" partner pair. Because there is only one "local" partner and many "remote" partners the partnership agreements are configured under the "remote" partner in the B2B configuration. For a bi-directional exchange there will be two partnership agreements associated with a "remote" partner, one for the inbound direction and one for the outbound direction. The exception to this is the HL7 Immediate ACK, which is implicitly generated by the B2B and does not require a partnership agreement to be defined. Configuring a Functional ACK on the inbound partnership agreement, on the other hand, requires a separate partnership agreement which defines the ACK.

Use the B2B Web Console to configure the identifiers, which the client will use to identify the "self" partner to which is will be sending, in the "self" partner profile. Set the "HL7 Application Id" to the value of "A19ADR" and the "HL7 Facility Id" to the value of "GWYQ".



Save the changes.

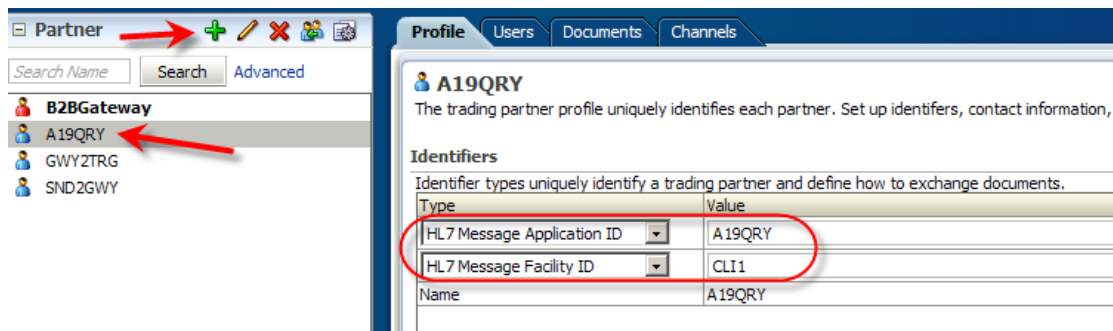
Note that the "self" partner already "knows" about the new document definitions. All we need to do, which is optional anyway, is configure the "direction" in which the documents of this type will be exchanged. Click the "Documents" Tab.



Uncheck "Sender" checkbox next to the QRY document and the "Receiver" checkbox next to the ADR document. The "remote" partner will be sending QRY documents and not receiving them. The "self" partner will be sending ADR document and not receiving them. Save the changes.

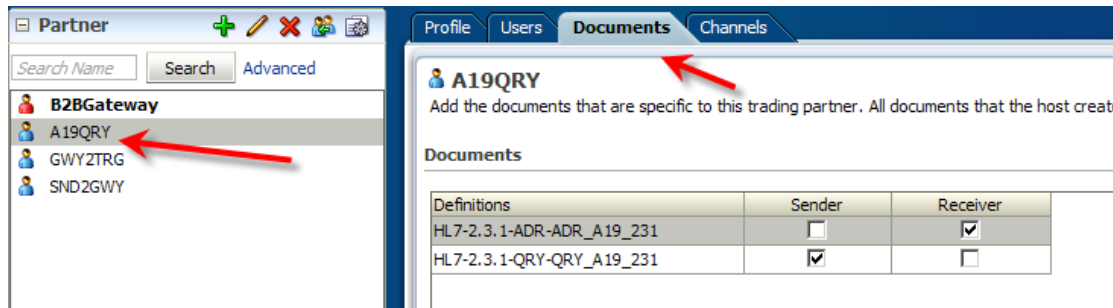
Create and Configure "Remote" Partner

Create a new Partner, named A19QRY and add two identifiers to the "Profile" - "HL7 Message Application ID": A19QRY and "HL7 Message Facility ID": CLI1.

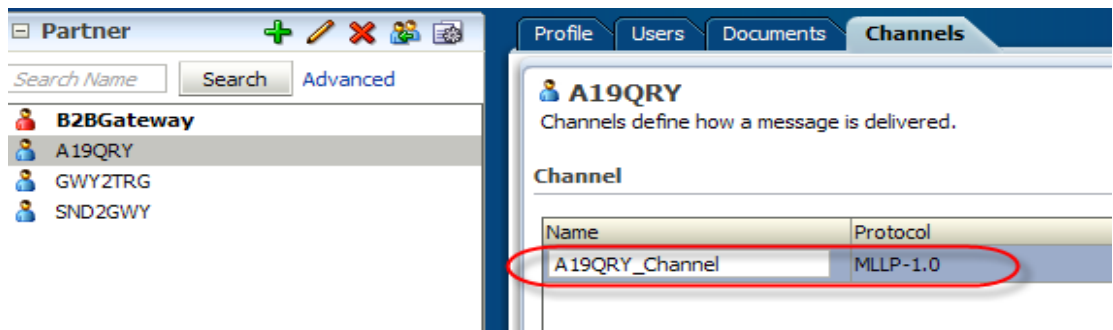


These are the identifiers the client will use as MSH-3 and MSH-4 in the QYR messages it will be sending.

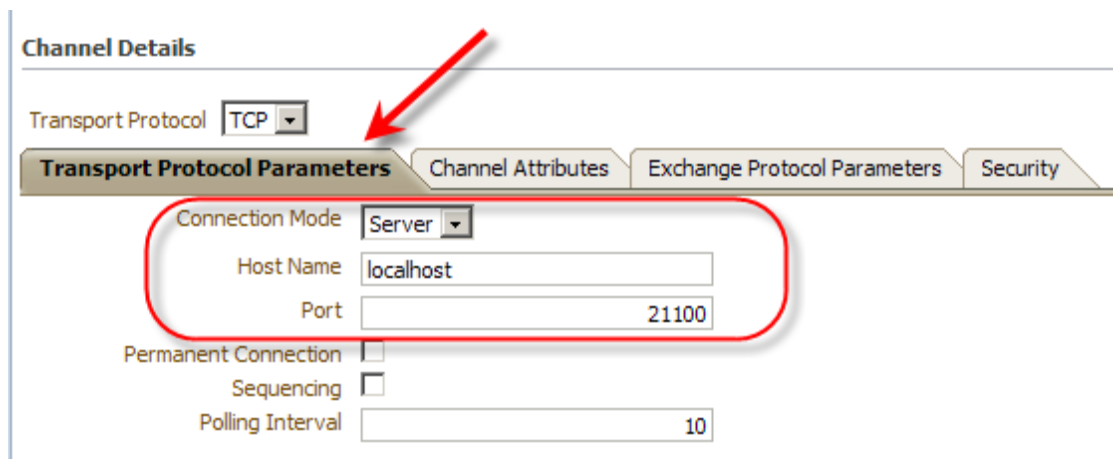
Add the two documents created earlier as documents that will be exchanged with this partner, configuring "direction" of exchange so that the QRY is being sent from the remote partner and ADR is being received by the remote partner.



Add a new MLLP-1.0 channel.



Configure the "Transport Protocol Parameters" as "Connection Mode": server, "Host Name": localhost and "Port Number": 21100.



Configure "Channel Attributes" as "Enable Channel": selected

Channel Details

Transport Protocol **TCP**

Transport Protocol Parameters | **Channel Attributes** | Exchange Protocol Parameters | Security

Ack Mode **None**

Retry Interval

Retry Count

Description

Enable Channel

Disable Channel

Compressed

Transport Callout

Configure "Exchange Protocol Parameters" as "Identify TP by delivery channel": checked.

Exchange Protocol Parameters | Security

End Block Character

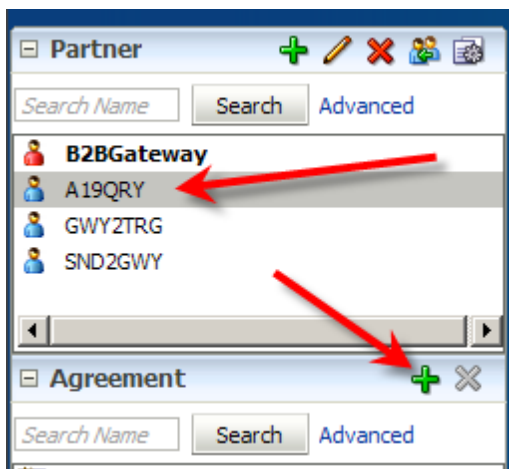
Carriage Return Character

Identify TP by delivery channel

"Save" your changes.

Configure Inbound QRY Agreement

Select the new partner in the "Partner" pane and add a new "Agreement" to the Agreement pane.



Give this agreement the Id of "A19QRY_231_In" and the name of "A19QRY_231_In_Agr".

Configure the document to exchange as "QRY_A19_231" in the direction from the "remote" to the "local" partner.

Agreement

A19QRY_231_In_Agr

B2BGateway ← QRY_A19_231 → A19QRY

Details

* Agreement Id: A19QRY_231_In

Name: A19QRY_231_In_Agr

Description: []

Start Date: []

End Date: []

Callout: []

Configure "Agreement Parameters": "Translate": checked and "FA Handled by B2B": No.

Agreement Parameters

Validate:

Translate:

Functional Ack:

FA Handled by B2B: No

Document Retry Interval: []

Document Retry Count: []

Configure "A19QRY" side by selecting the "Channel" and the "Identifiers".

A19QRY

Channel: A19QRY_Channel

Identifiers

Type	Value
HL7 Message Application ID	A19QRY
HL7 Message Facility ID	CLI1
Name	A19QRY

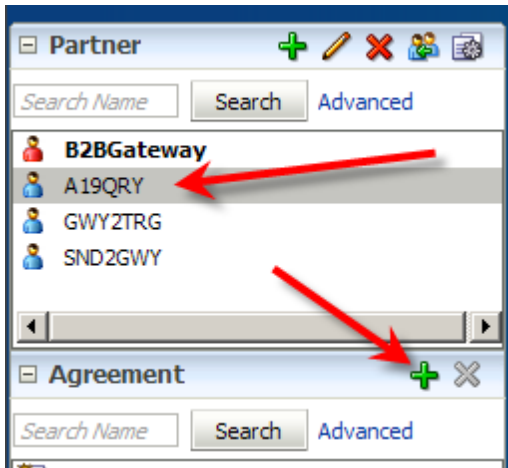
Save and Deploy the agreement.

B2BGateway ← QRY_A19_231 → A19QRY

Save Validate Deploy

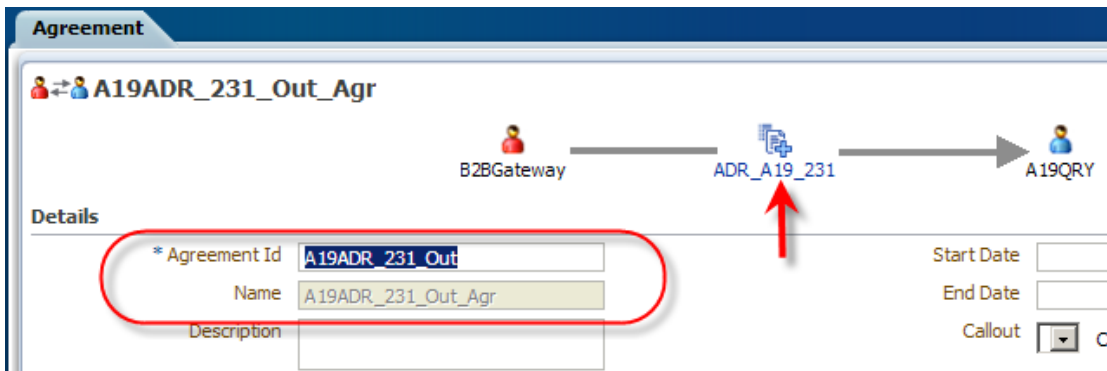
Configure Outbound ADR Agreement

Select the new partner in the "Partner" pane and add a new "Agreement" to the Agreement pane.



Give this agreement the Id of "A19ADR_231_Out" and the name of "A19ADR_231_Out_Agr".

Configure the document to exchange as "ADR_A19_231" in the direction from the "local" to the "remote" partner.

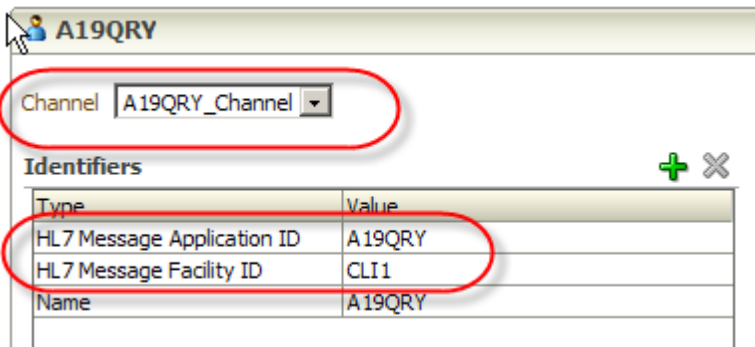


Configure "Agreement Parameters": "Translate": checked.

Agreement Parameters

- Validate
- Translate
- Functional Ack

Configure "A19QRY" side by selecting the "Channel" and the "Identifiers".



Save and Deploy the agreement.

Verify and Set Global B2B Configuration

To make sure, verify that specific global B2B configuration settings are as required.

Administration-->Configuration

Acknowledgement

Functional ACK Handled by B2B: TRUE

Miscellaneous

Generic Message Type: TRUE

Miscellaneous(Continued)

Outbound Dispatcher Count: 1

Inbound Dispatcher Count: 1

Auto Stack Handler Interval: 1

Non Purgeable

Use JMS Queue as Default: TRUE

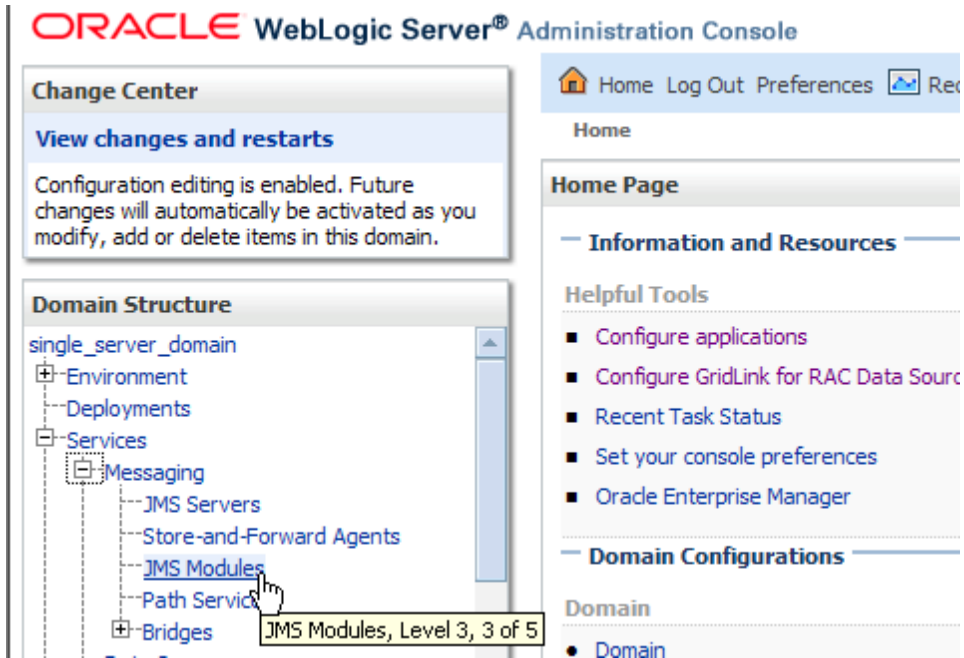
Save changes if you made changes.

SOA Composite Configuration

Create a B2BA19Queue

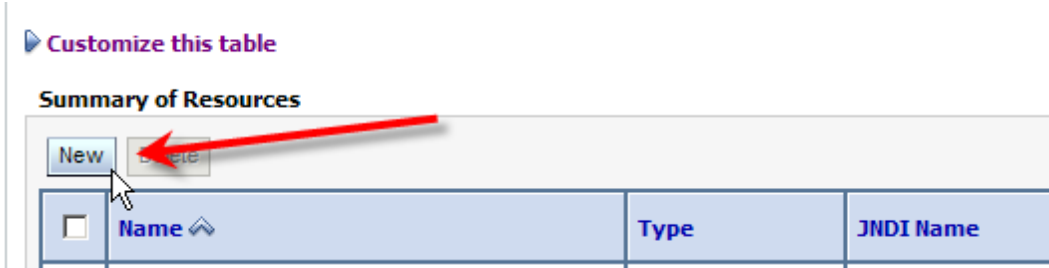
The A19 query processor will be a single SOA Composite Application with two projects, the project which receives the QRY message and the project which sends the ADR response. The reason for braking up this solution into two projects is obscure and has to do with XSD namespace conflict between the QRY XSD and the ADR XSD, as generated by the B2B Document Editor. I make sure that each project uses either one or the other of the XSDs but not both. The two projects are integrated using a JMS Queue. It is necessary to use a JMS Queue because B2B properties must be propagated from the sender to the receiver and the only reasonably efficient adapter which is capable of passing metadata is the JMS Adapter. If the namespace conflict was not there, there would be no need to two projects and no need for a JMS Queue to connect them together.

Start the WebLogic Server Console web application, typically <http://localhost:7001/console>, and select Services --> Messaging --> JMS Modules



Click on the SOAJMSModulke link.

Click on the New button.



Choose Queue radio button and click Next button.

Create a New JMS System Module Resource

Back Next Finish Cancel

Choose the type of resource you want to create.

Use these pages to create resources in a JMS system module, such as queues, topics, temp...

Depending on the type of resource you select, you are prompted to enter basic information, connection factories, distributed queues and topics, foreign servers, and JMS SAF destinations. You can also associate targetable resources with subdeployments, which is an advanced mechanism.

Connection Factory

Queue

Enter "B2BA19Queue" for queue name, "jms/B2BA19Queue" for JNDI name and click Next.

Create a New JMS System Module Resource

Back Next Finish Cancel

JMS Destination Properties

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

* Indicates required fields

* **Name:**

JNDI Name:

Template:

Back Next Finish Cancel

Select SOASubDeployment from the drop-down of subdeployments, ensure the SOAJMSServer radio button is selected and click Finish.

Create a New JMS System Module Resource

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

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

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

Subdeployments:

What targets do you want to assign to this subdeployment?

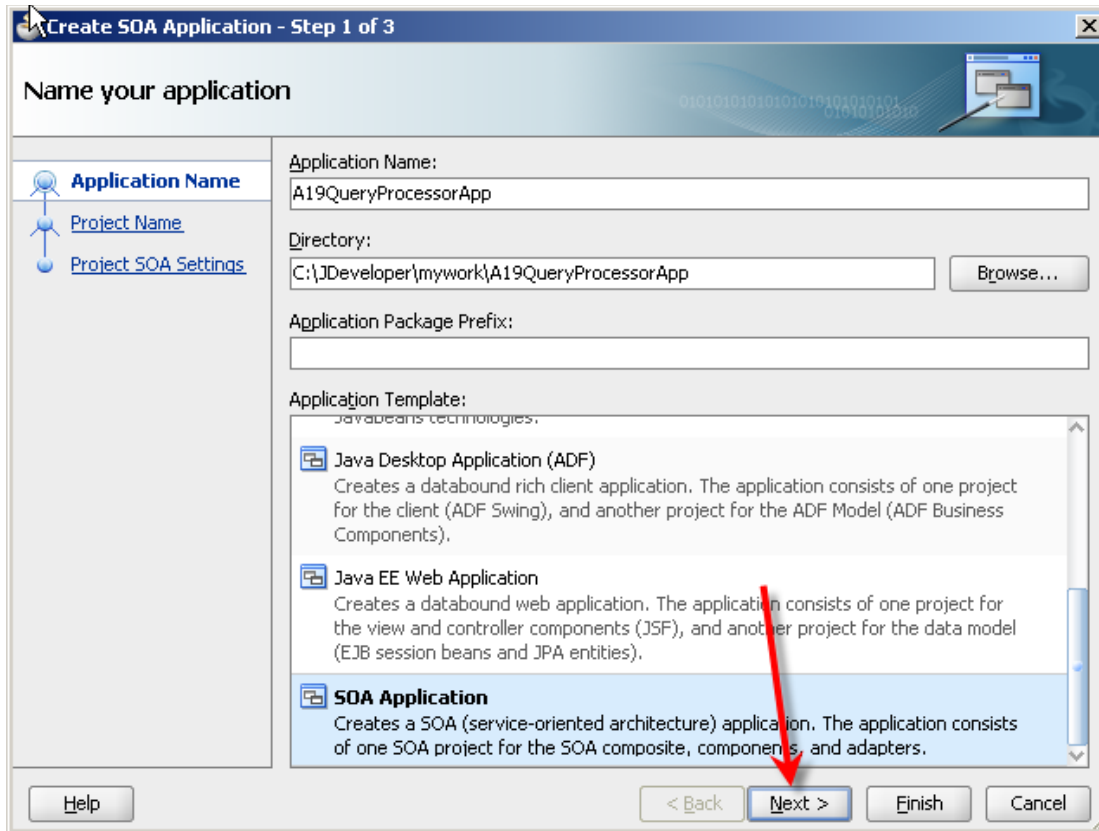
Targets :

JMS Servers
<input type="radio"/> BPMJMSServer
<input type="radio"/> JRFWSAsyncJmsServer
<input checked="" type="radio"/> SOAJMSServer

The JMS Queue is now created and ready for use. Close the WebLogic Admin Console.

A19Processor Application

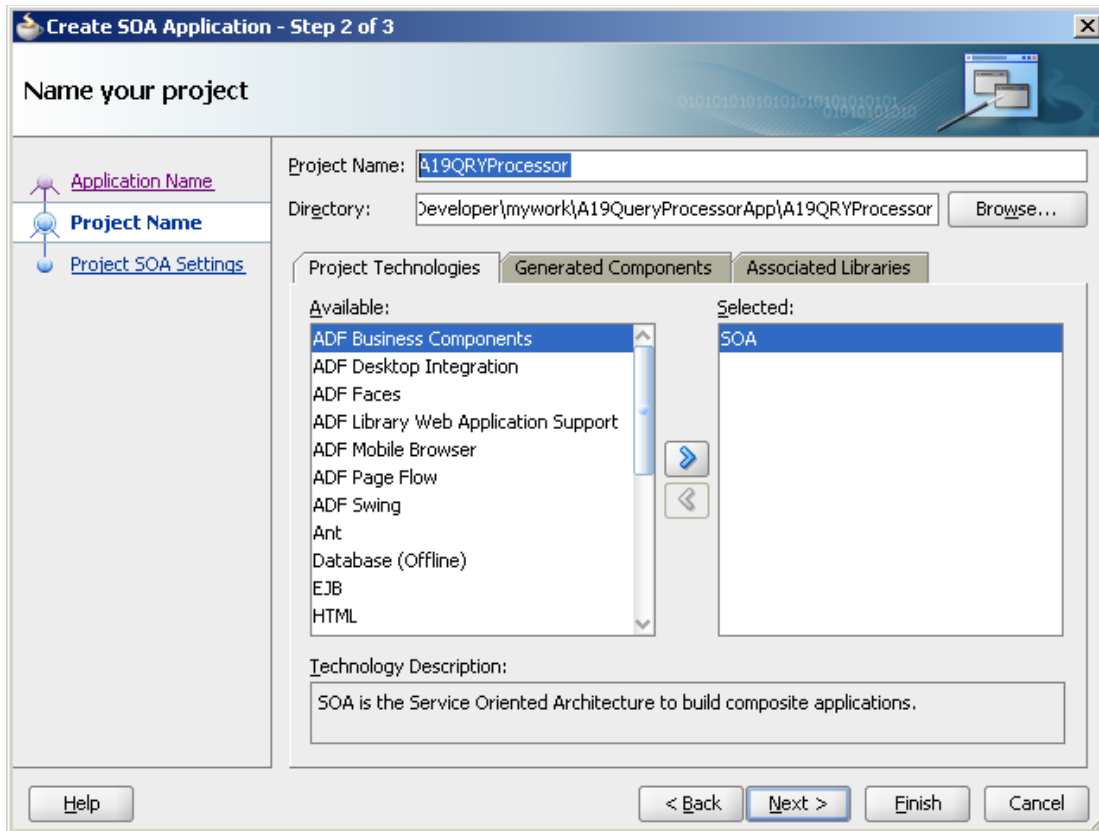
Start JDeveloper and create a new SOA Application, A19QueryProcessorApp - click Next on the initial Dialogue.



QRY Receiver Project

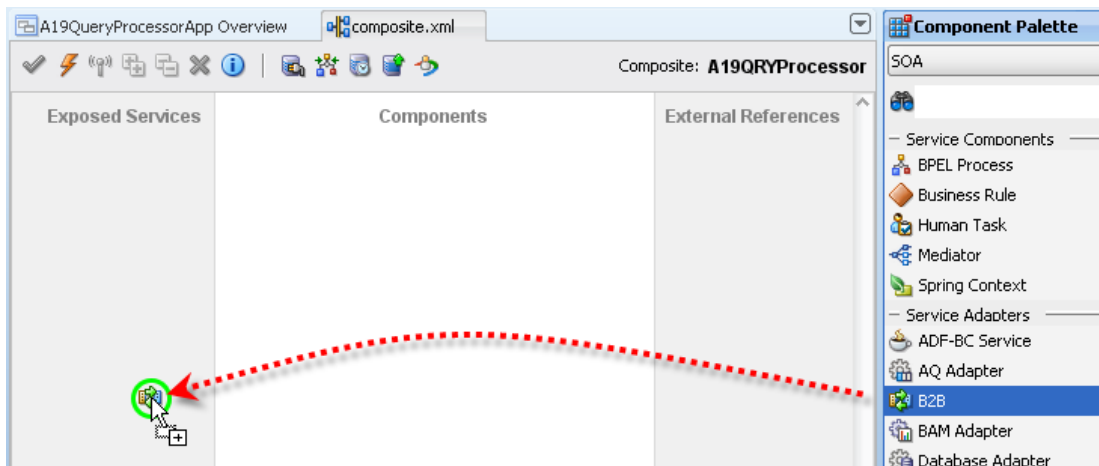
The first of the two projects will receive a QRY message from the B2B Adapter, will map it to the non-B2B QRY XSD, set JMS properties to propagate b2b metadata and send the message to the JMS Queue using the JMS Adapter.

Name the new project "A19QRYProcessor", make sure it is a SOA project and click Next.



Leave "Empty Composite" selected and click Finish.

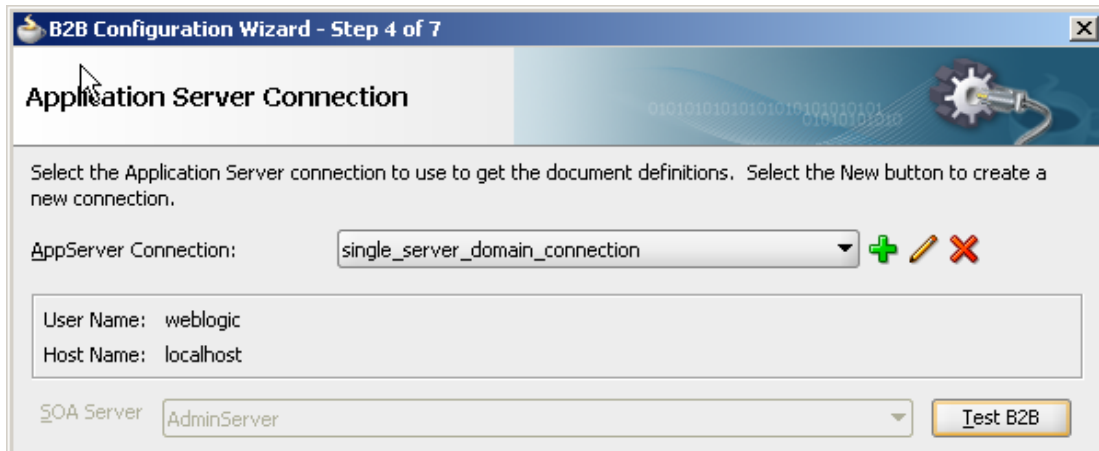
Drag the B2B component form the "Service Adapters" list to the "Exposed Services" swim line.



Name the service B2BQRYIn.

Leave B2B Integration Type at default.

Choose your AppServer Connection and Test B2B to make sure it works.

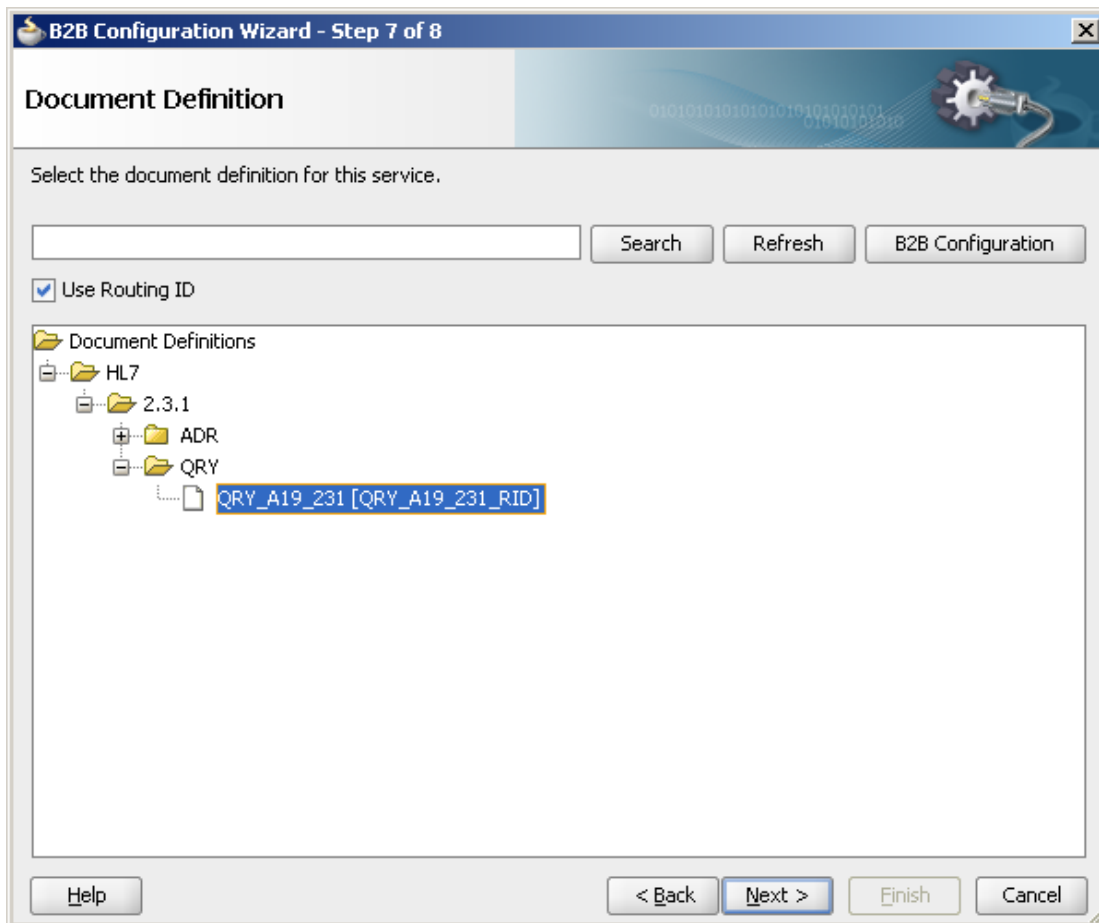


Choose "Receive" operation.

Leave Document Definition Handling at Basic (default).

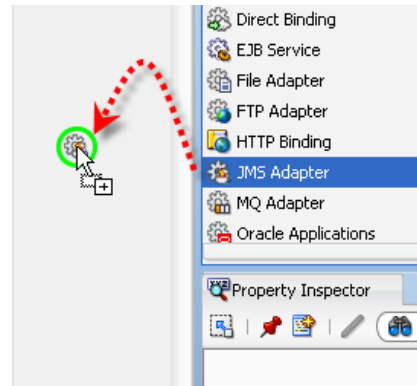
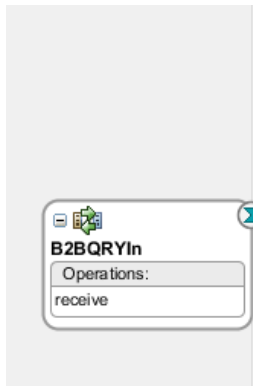
Click "Document Routing ID".

Choose HL7-->2.3.1-->QRY-->QRY_A19_231 document type associated with the Routing ID.



Finish.

Drag the JMS Adapter component from the Service Adapters list to the External References swim line.



Name the service JMSQRYOut.

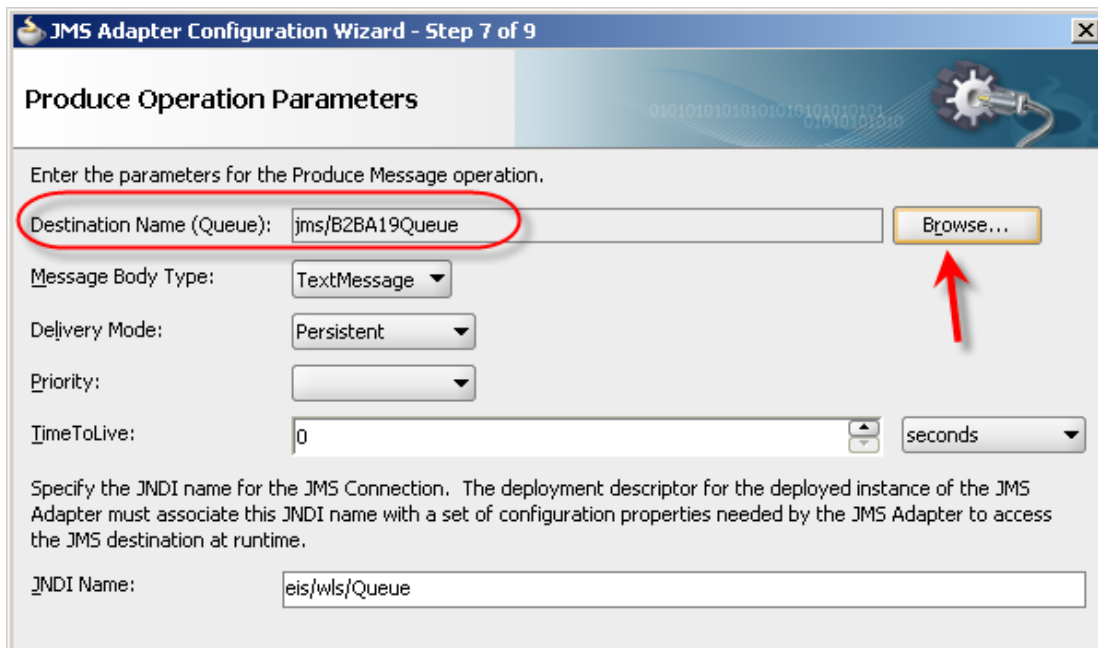
Choose WebLogic JMS as the Oracle Enterprise Messaging Service.

Choose your AppServer Connection.

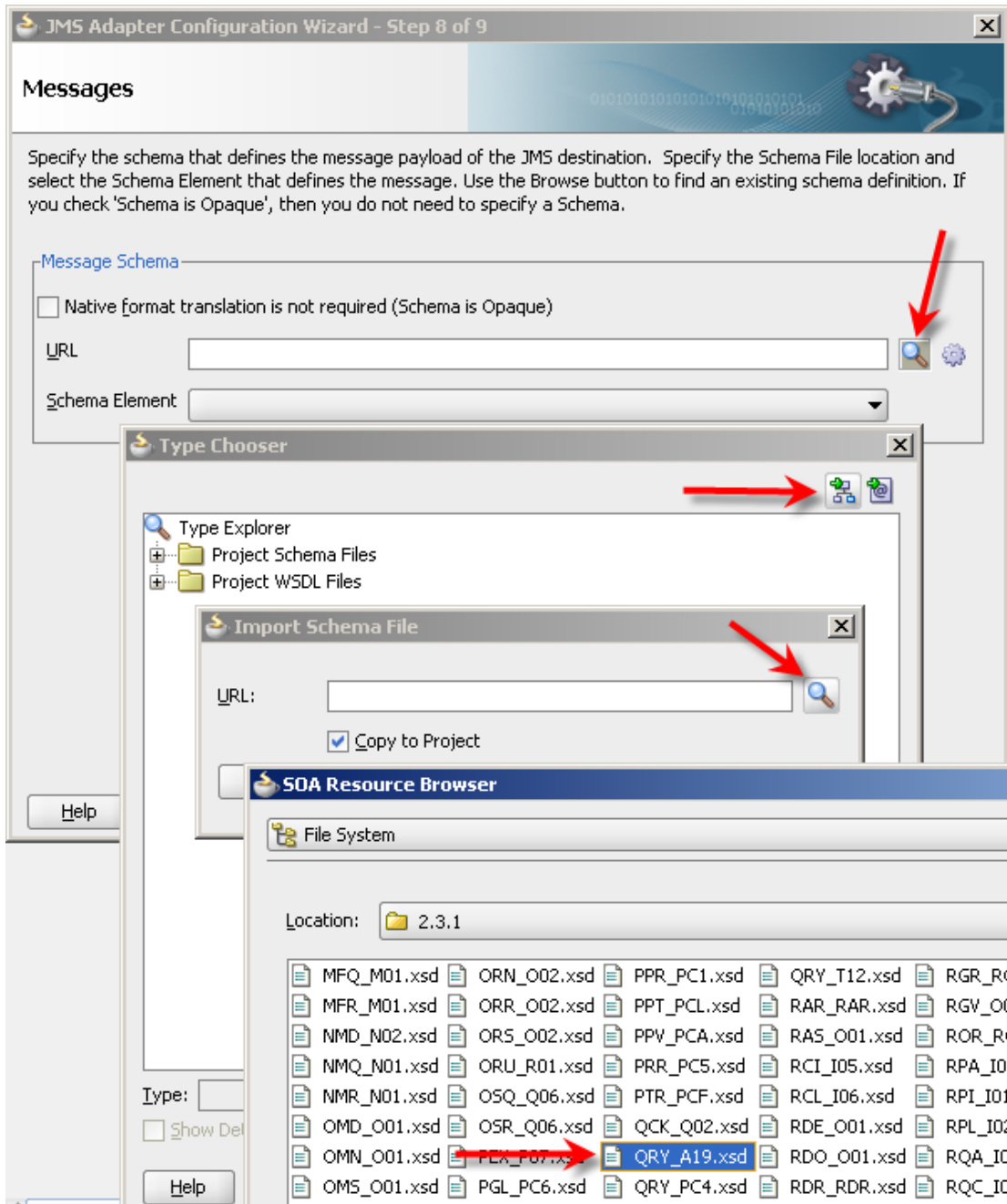
Leave Adapter Interface at default (specified later).

Choose Produce Message as operation type.

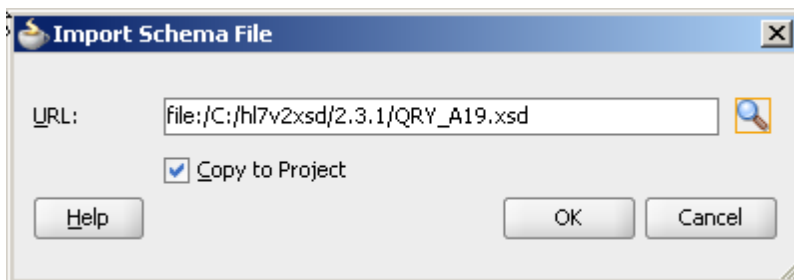
Choose jms/B2BA19Queue as the queue to use.



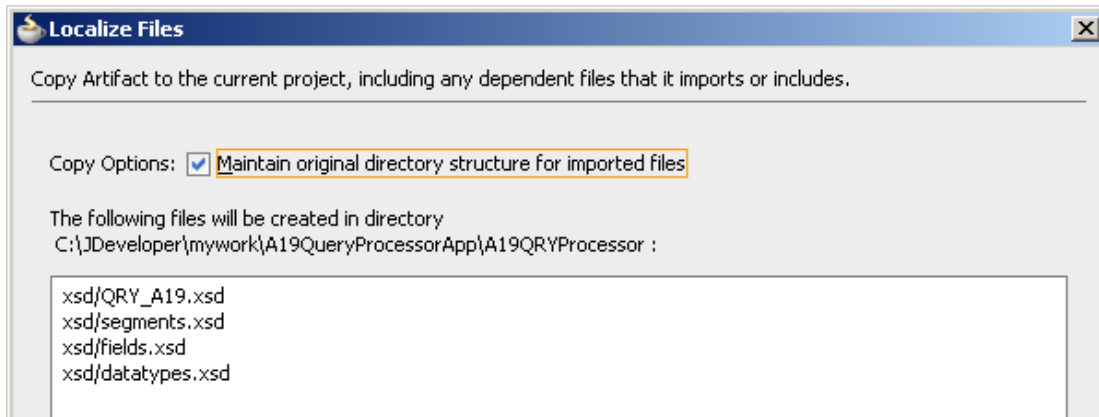
Locate the QRY_A19.xsd, from the hl7v2xsds.zip distribution, which you downloaded and expanded earlier, and choose it.



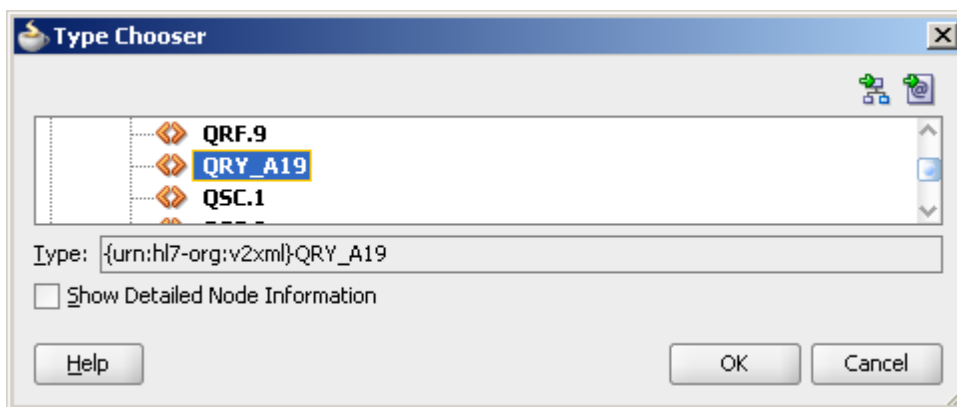
Select Save copy to the project and click OK.



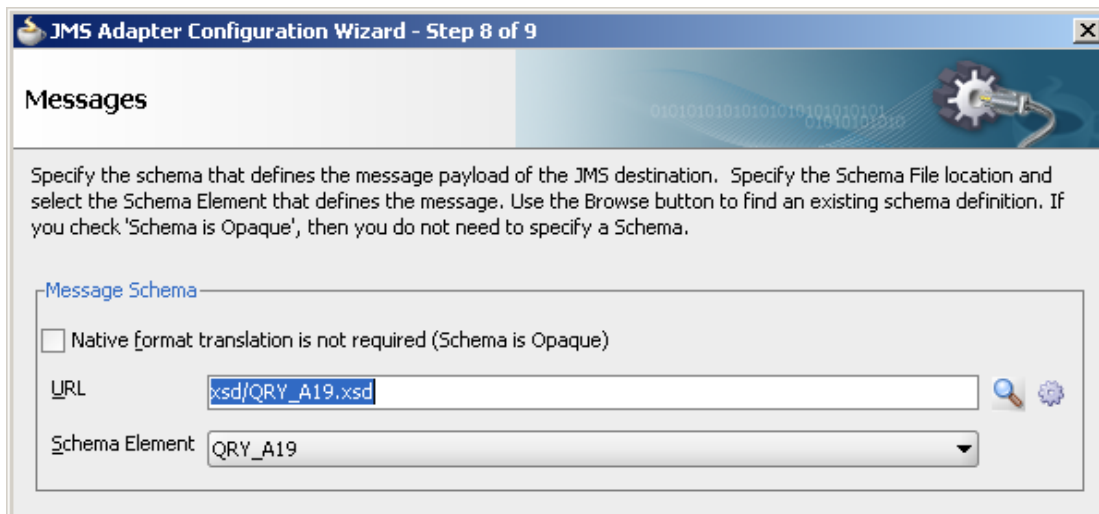
Accept defaults and click OK.



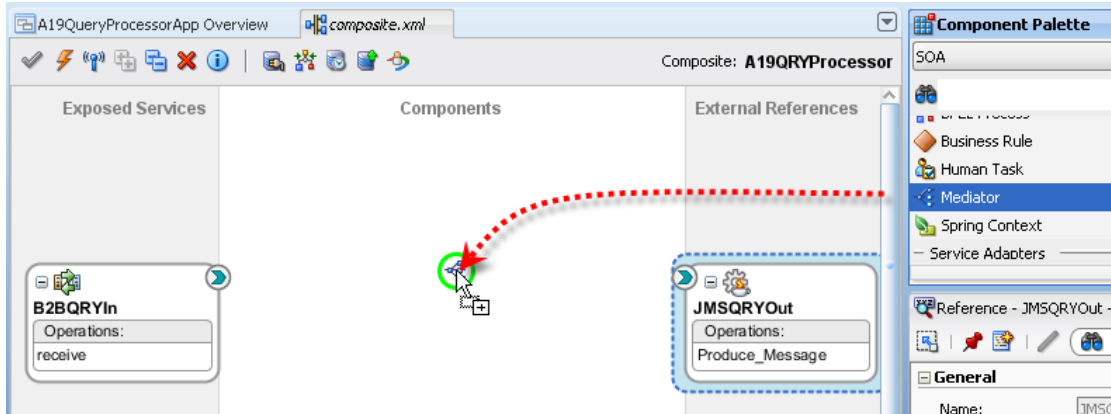
Choose the QRY_A19 Element and click OK.



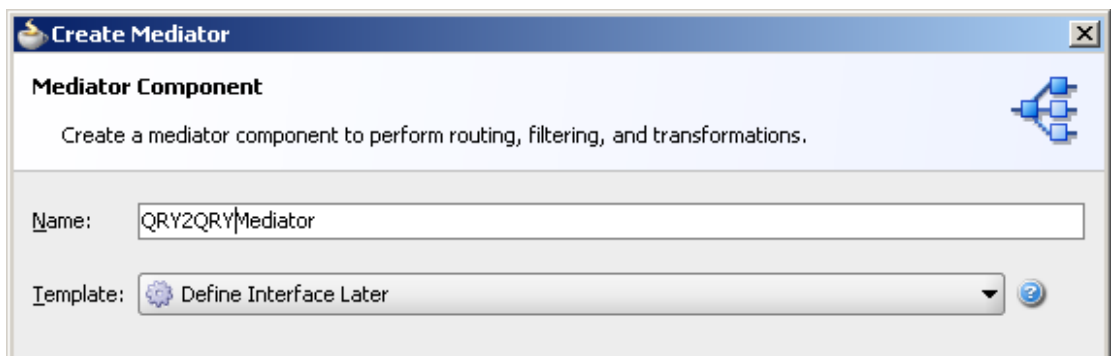
Complete the wizard by clicking Next and Finish.



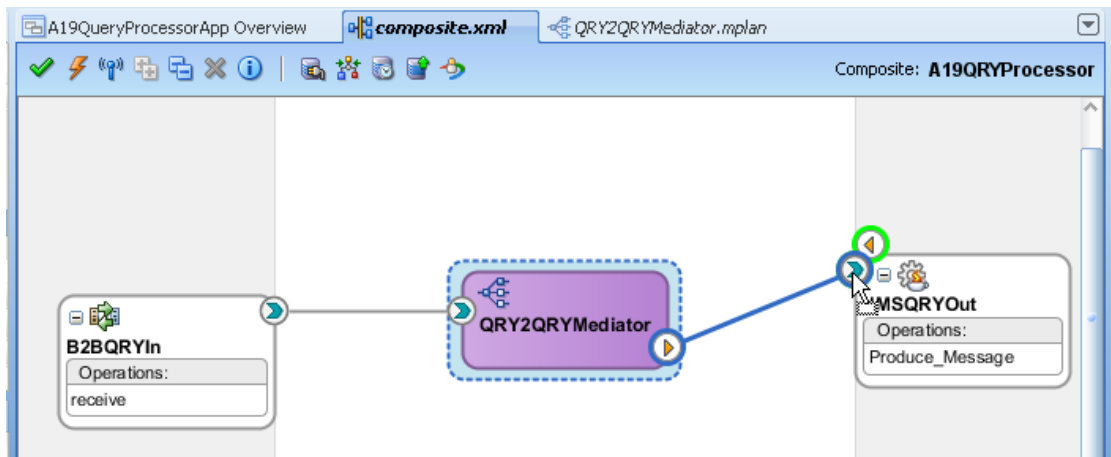
Drag the Mediator component onto the Components swim line.



Name the Mediator component QRY2QRYMediator and accept the default "Define Interface Later" Template.

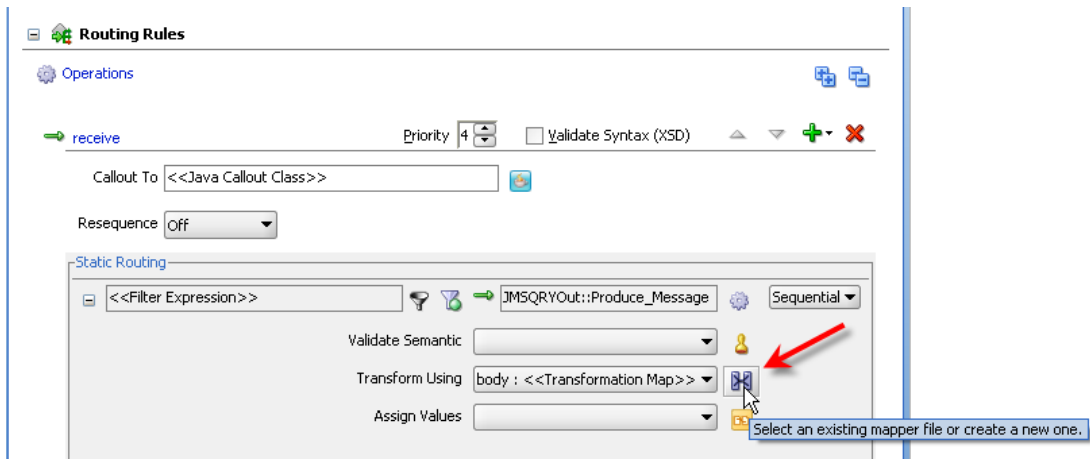


Connect the B2BQRYIn Adapter to the QRY2QRYMediator and QRY2QRYMediator to JMSQRYOut Adapter.

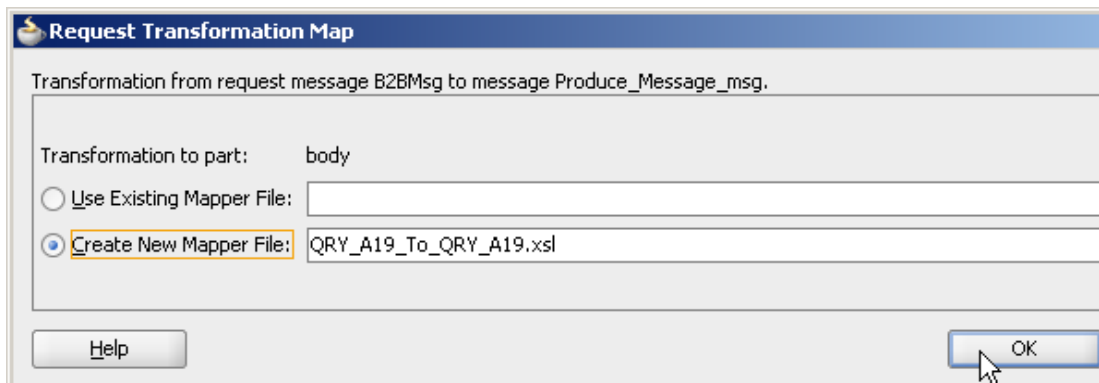


Double-click the QRY2QRYMediator top open its mplan.

Click the Mapper button to create a new mapper file.



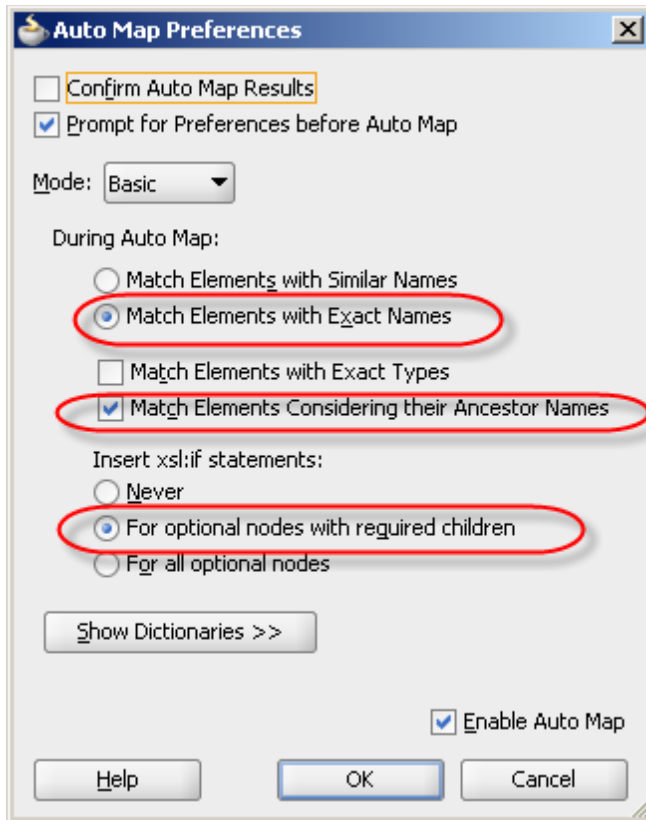
Select the Create New Mapper File radio button and click OK.



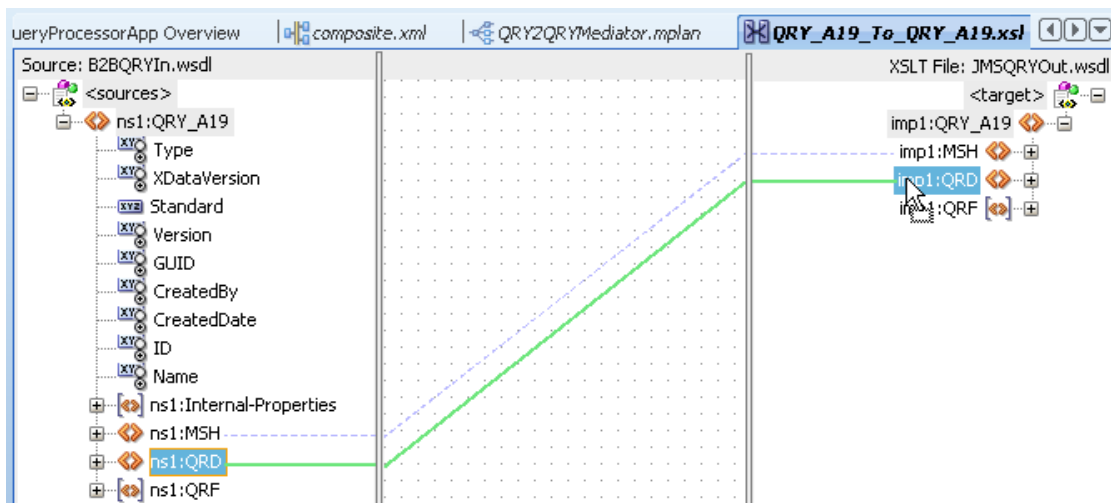
Drag from MSH segment at the left to the segment MSH at the right to start the mapping wizard.



Ensure the dialogue is configured as indicated and click OK.

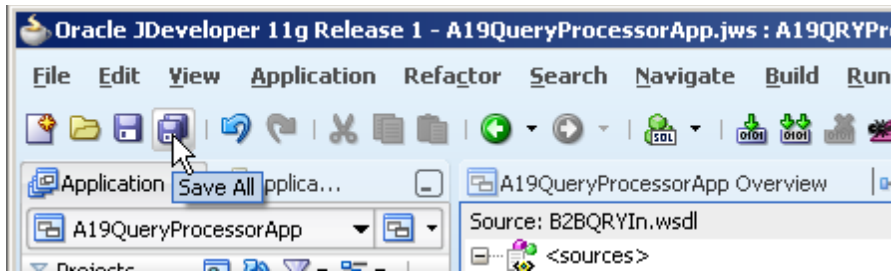


Repeat the process for the QRD segment.

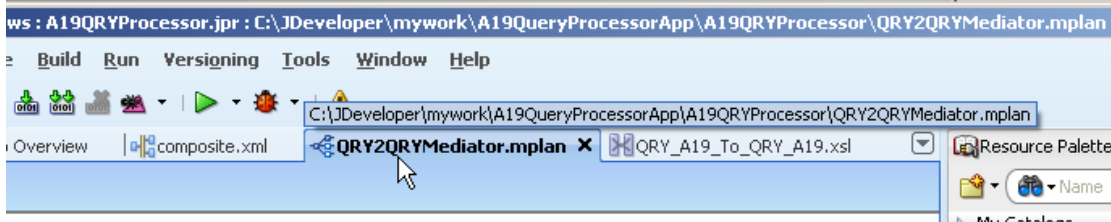


Click the Save All toolbar button.

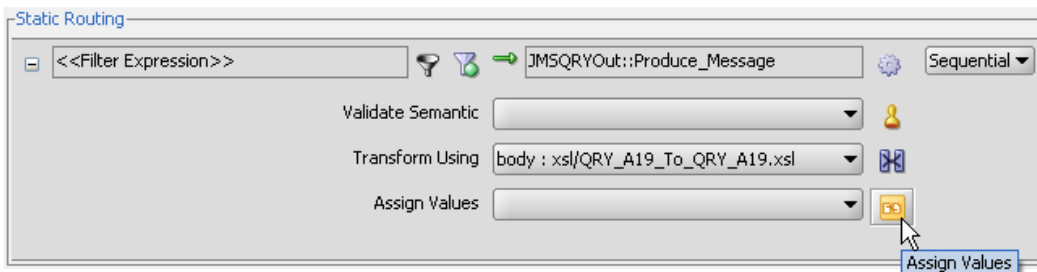
We created a mapping between the QRY message received from the B2B and the intermediate QRY message with a different namespace to that the B2B Document Editor produced when exporting the QRY and ADR Messages. Recall from the earlier discussion that this is a workaround for the namespace conflict issue discussed earlier.



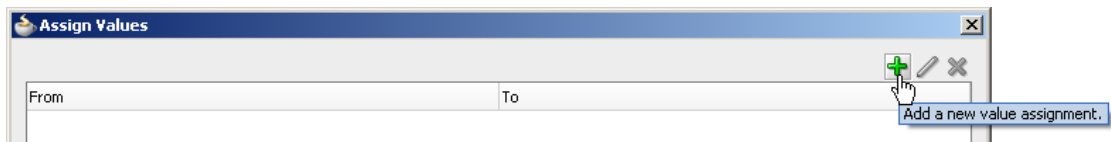
Switch to QRY2QRYMediator.mplan



Click the "Assign Values" button.

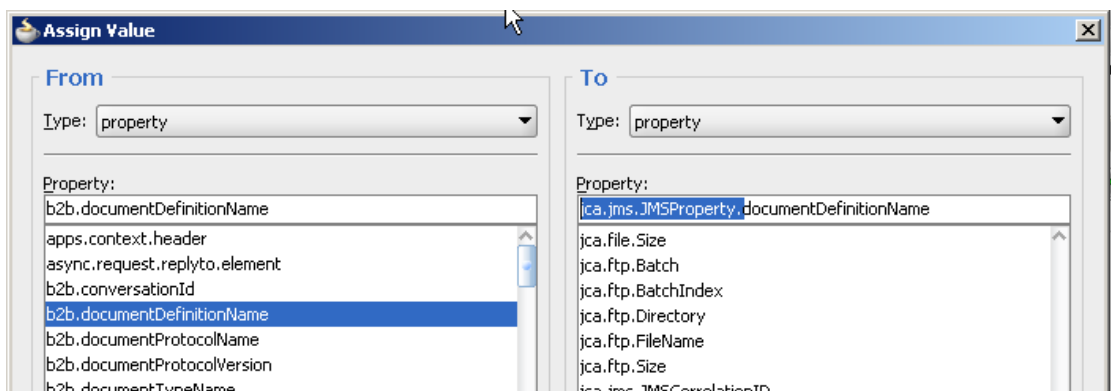


Click the "Add new value assignment" button.

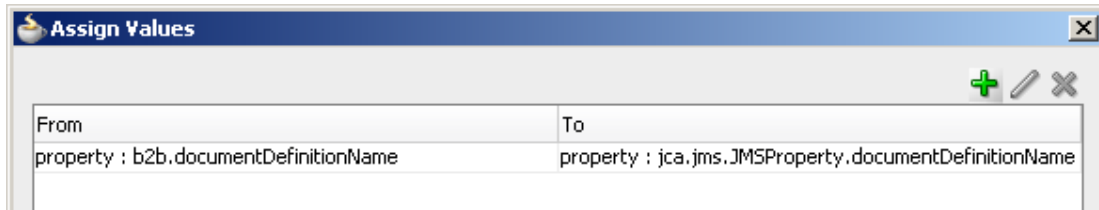


Select the b2b.documentDefinitionName in the left hand properties list.

Enter jca.jms.JMSProperty.doumentDefinitionName property in the right hand property data entry box and click OK.



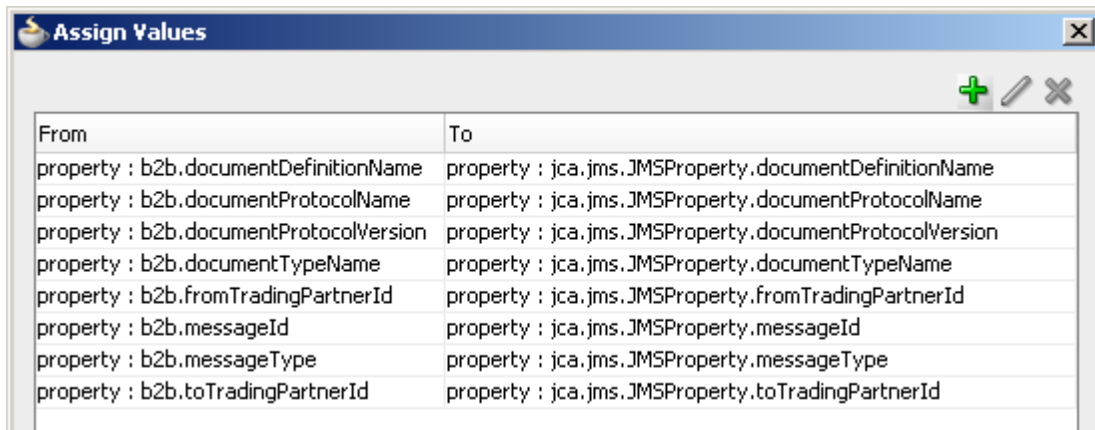
The Assign Values dialogue box will show the new assignment.



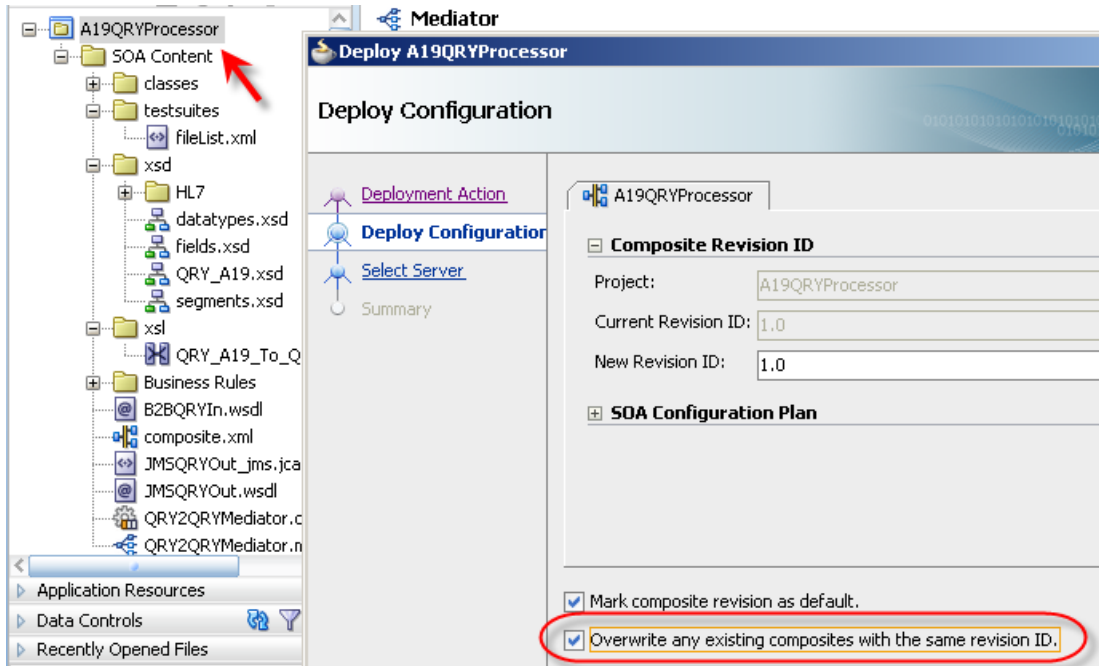
Repeat the process to assign b2b properties to JMS properties as shown in the table below.

b2b.documentProtocolName	jca.jms.JMSProperty.documentProtocolName
b2b.documentProtocolVersion	jca.jms.JMSProperty.documentProtocolVersion
b2b.documentTypeName	jca.jms.JMSProperty.documentTypeName
b2b.fromTradingPartnerId	jca.jms.JMSProperty.fromTradingPartnerId
b2b.messageId	jca.jms.JMSProperty.messageId
b2b.messageType	jca.jms.JMSProperty.messageType
b2b.toTradingPartnerId	jca.jms.JMSProperty.toTradingPartnerId

The completed Assign Values dialogue should look like that shown below.



Right-click on the name of the project, choose Deploy and follow the wizard to deploy this project.



Check the deployment log to make sure the project was built and deployed.

```

[05:45:27 PM] Preparing to send HTTP request for deployment
[05:45:27 PM] Creating HTTP connection to host:xp64.au.oracle.com, port:7001
[05:45:27 PM] Sending internal deployment descriptor
[05:45:27 PM] Sending archive - sca_A19QRYProcessor_rev1.0.jar
[05:47:27 PM] Received HTTP response from the server, response code=200
[05:47:27 PM] Successfully deployed archive sca_A19QRYProcessor_rev1.0.jar to partition "default" on server AdminServer
[05:47:27 PM] Elapsed time for deployment: 2 minutes, 24 seconds
[05:47:27 PM] ---- Deployment finished. ----

```

The first half of the solution, one which receives QRY messages and deposits them in a JMS Queue with properties to carry B2B metadata, is completed and deployed.

ADR Sender Project

The second of the two projects will actually produce the ADR response. It will receive the non-B2B QRY message and use its values and some literal values to construct the ADR response. It will also set the B2B properties using the JMS properties propagated from the B2B receiver via the first project. These properties will enable the B2B outbound infrastructure to work out which partnership agreement to use and how to correlate this message to the original request.

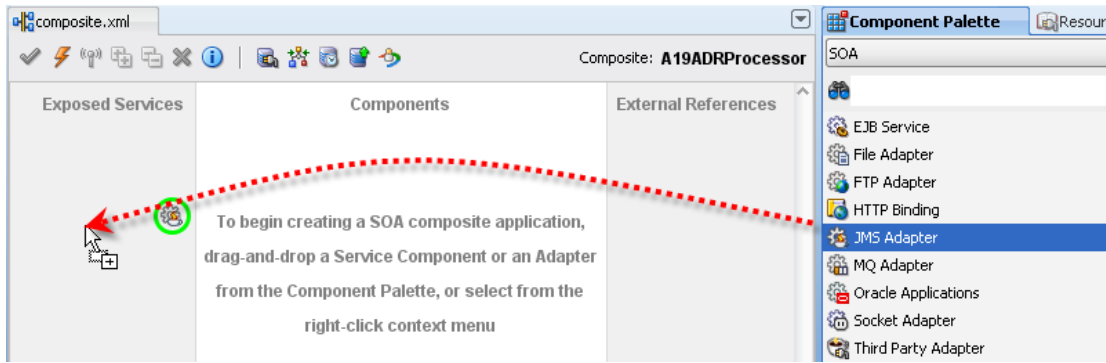
Pull down JDeveloper menu "File" --> "New...".

Select "SOA Tier" --> "SOA Project" and click "OK".

Name the project "A19ADRProcessor" and click "Next".

Accept default "Empty Composite" and click "Finish".

Drag the "JMS Adapter" service adapter from the SOA Component Palette to the "Exposed Services" swim line.



Name the new service "JMSQRY2ADRIn".

Choose "WebLogic JMS" and the Oracle Enterprise Messaging Service.

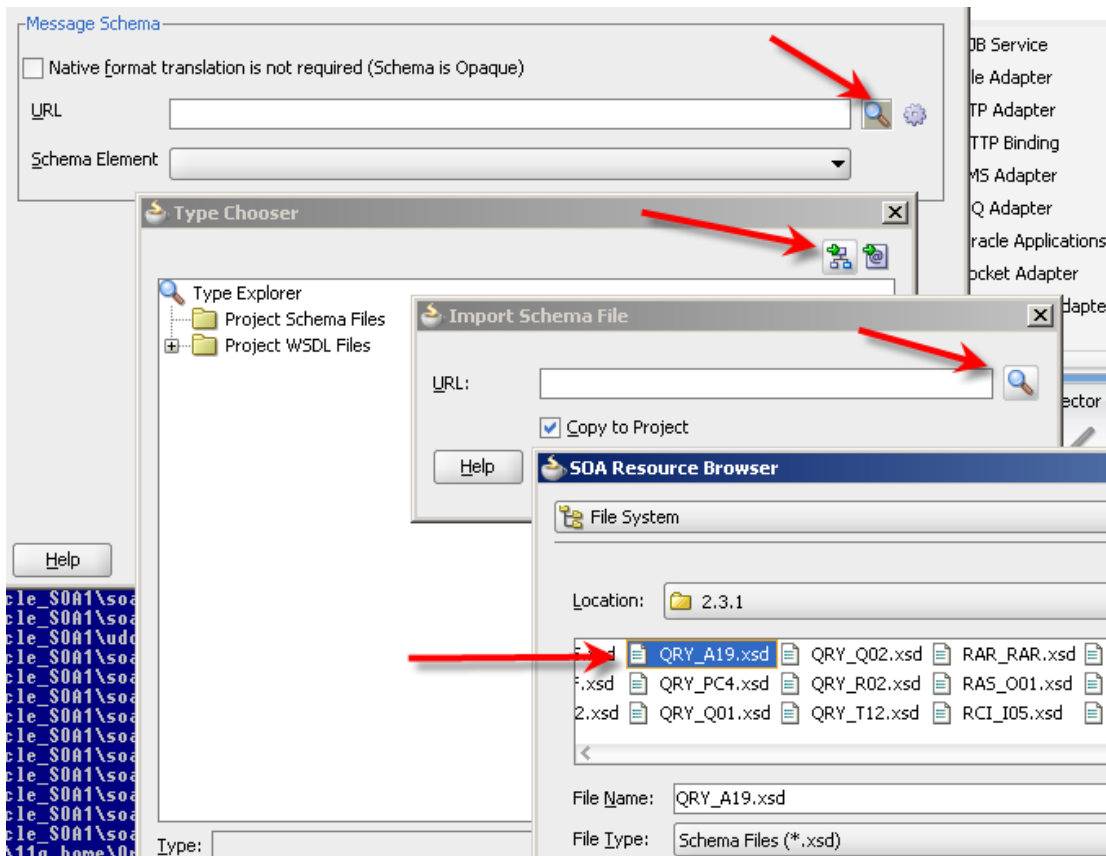
Choose your AppServer Connection.

Accept the default Adapter Interface (specified later).

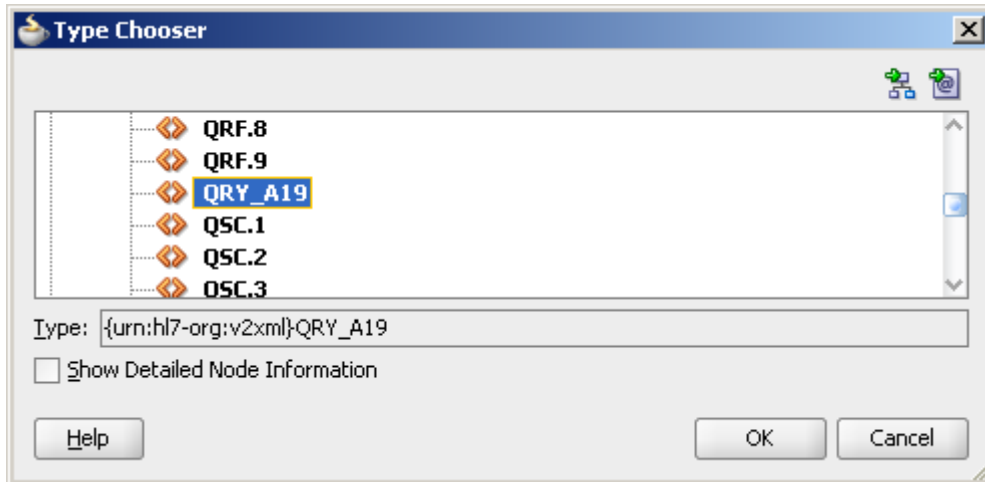
Choose "Consume Message" operation.

Browse for and select the "B2BA19Queue" JMS Queue.

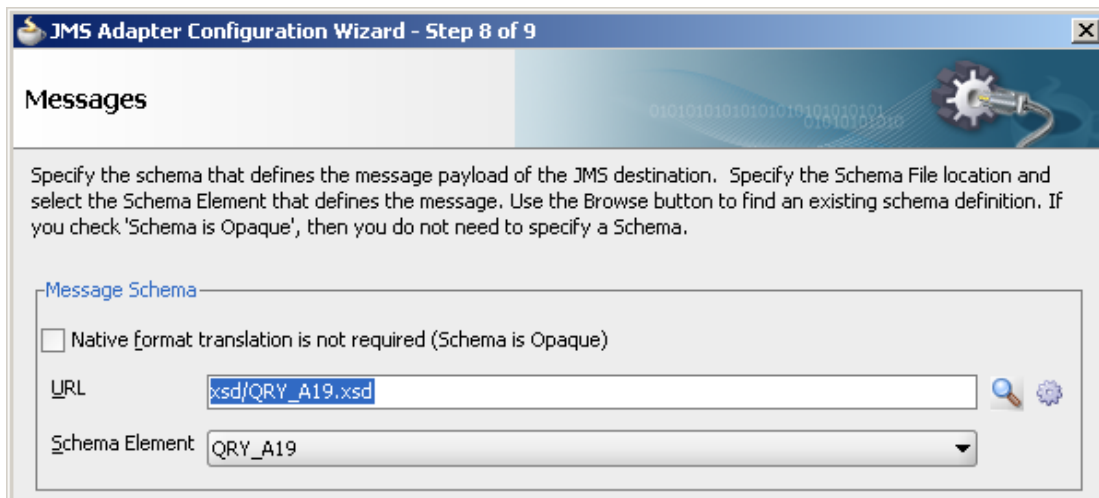
Browse for and Import the ARY_A19.xsd schema from the hl7v2xsd schemas you downloaded and unarchived earlier.



Using Type Chooser locate the QRY_A19 Element in the imported schema and select it as the element to use.

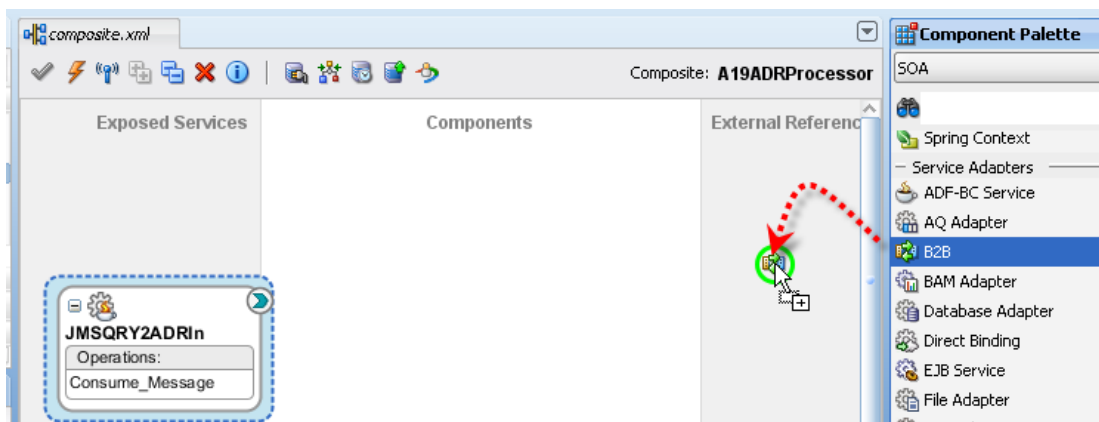


Accept the schema element.



Click Next and Finish.

Drag the B2B Adapter to the External Services swim line.



Name the service "B2BADROut".

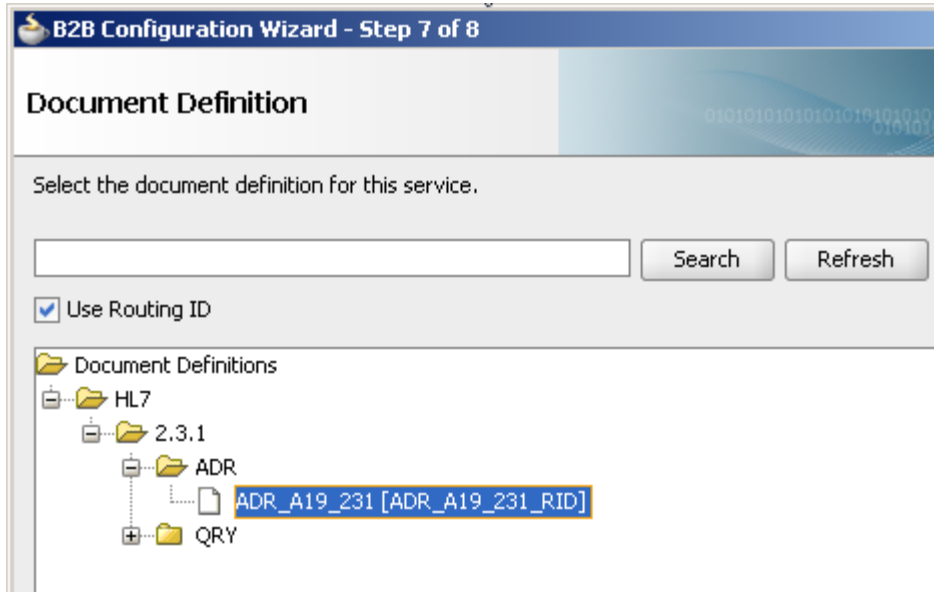
Accept default for "B2B Integration Type".

Choose your AppServer Connection.

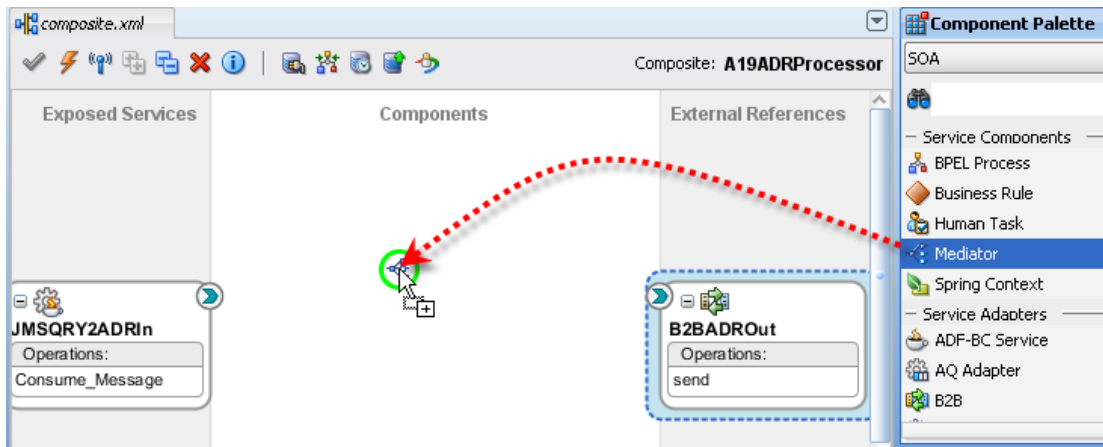
Accept "Send" operation.

Accept default Basic "Document Definition Handling".

Check "Use Routing ID", select the "ADR_A19_231" document definition and click Finish.

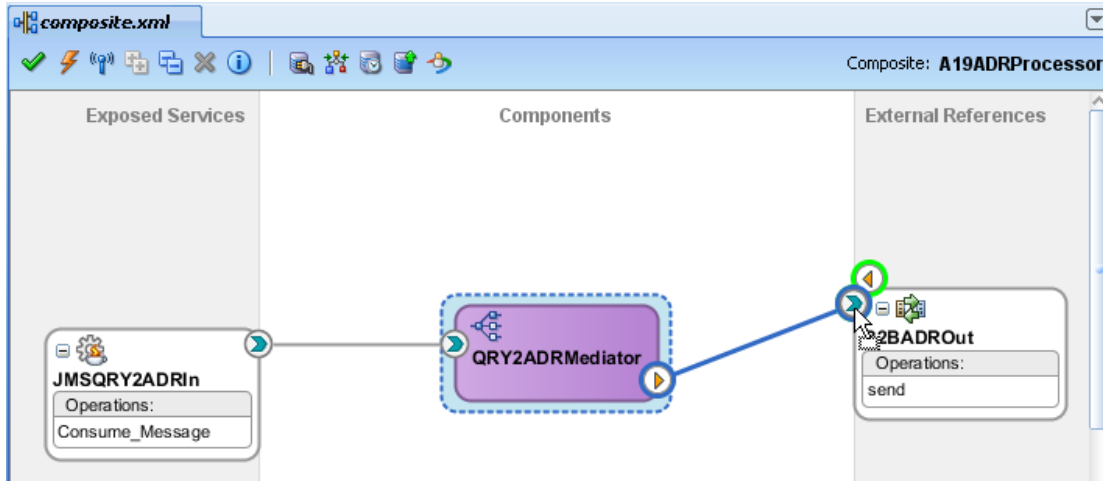


Drag the Mediator service component to the Components swim line.



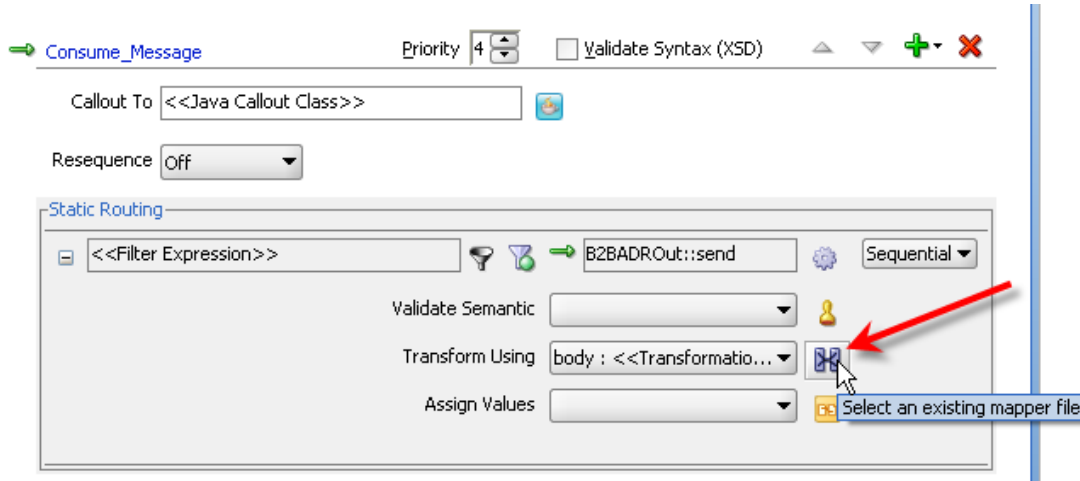
Name it "QRY2ADRMediator" and accept the default "define interface later" template.

Connect the components.

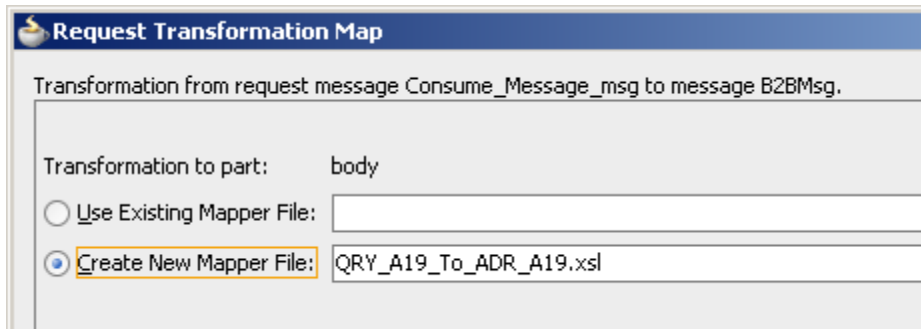


Double-click the QRY2ADRMediator component to open its mplan.

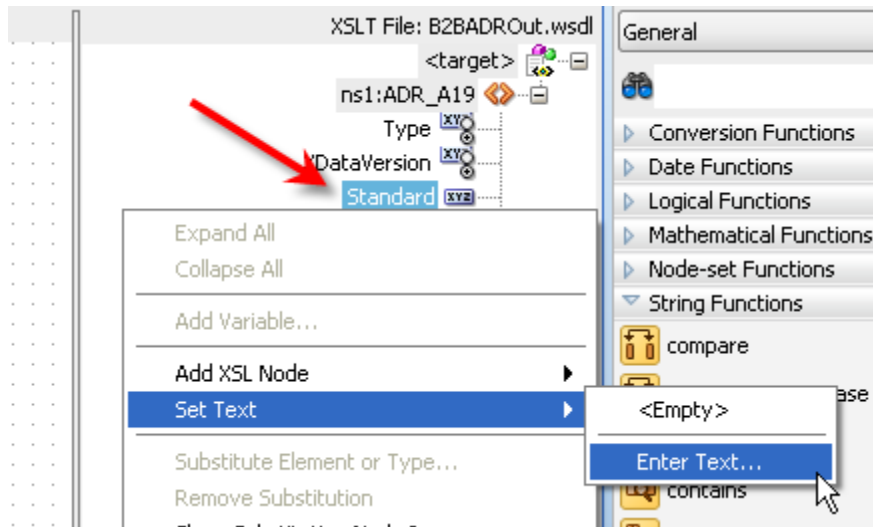
Click on the Mapper button.



Choose "Create New Mapper File" and click "OK".



Set text "HL7" for the Standard attribute of the ADR_A19 message on the left.



Configure outbound ADR_A19 structure by mapping from the inbound QRY_A19 structure, setting literal text values and configuring functions as follows:

From	To
QRY_A19-->MSH-->MSH.1	ADR_A19-->MSH-->MSH.1
QRY_A19-->MSH-->MSH.2	ADR_A19-->MSH-->MSH.2
QRY_A19-->MSH-->MSH.3	ADR_A19-->MSH-->MSH.5
QRY_A19-->MSH-->MSH.4	ADR_A19-->MSH-->MSH.6
QRY_A19-->MSH-->MSH.5	ADR_A19-->MSH-->MSH.3
QRY_A19-->MSH-->MSH.6	ADR_A19-->MSH-->MSH.4
QRY_A19-->MSH-->MSH.7	ADR_A19-->MSH-->MSH.7
ADR	ADR_A19-->MSH-->MSH.9-->MSG.1
A19	ADR_A19-->MSH-->MSH.9-->MSG.2
QRY_A19-->MSH-->MSH.10	concat('ACK_',ADR_A19-->MSH-->MSH.10)
QRY_A19-->MSH-->MSH.11	ADR_A19-->MSH-->MSH.11
QRY_A19-->MSH-->MSH.12	ADR_A19-->MSH-->MSH.12
AA	ADR_A19-->MSA-->MSA.1
QRY_A19-->MSH-->MSH.10	ADR_A19-->MSA-->MSA.2
QRY_A19-->QRD	ADR_A19-->QRD
QRY_A19-->QRD-->QRD.8-->XCN.1	ADR_A19-->QUERY_RESPONSE-->PID-->PID.3-->CX.1
QRY_A19-->QRD-->QRD.8-->XCN.9	ADR_A19-->QUERY_RESPONSE-->PID-->PID.3-->CX.4

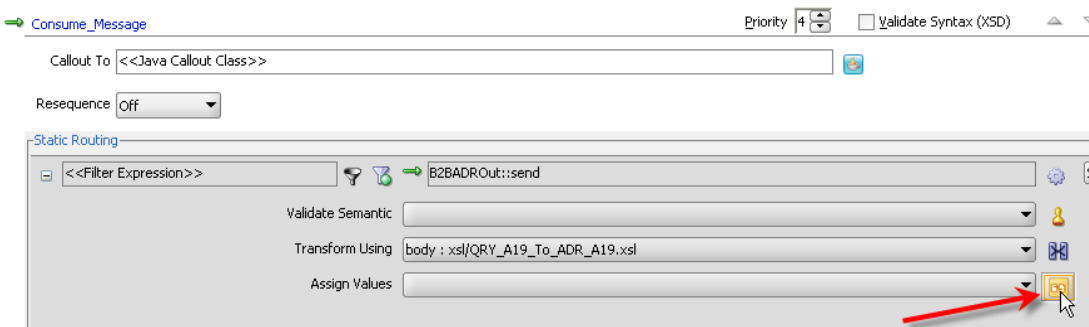
Smith	ADR_A19-->QUERY_RESPONSE-->PID-->PID.5-->XPN.1-->FN.1
John	ADR_A19-->QUERY_RESPONSE-->PID-->PID.5-->XPN.2
19610111	ADR_A19-->QUERY_RESPONSE-->PID-->PID.7-->TS.1
M	ADR_A19-->QUERY_RESPONSE-->PID-->PID.8
0908070605	ADR_A19-->QUERY_RESPONSE-->PID-->PID.19

Save the XSL Transform and close it.

This creates the transformation which will use data from the QRY message, and fixed values, to construct the ADR message. Normally one would have logic to access a database or something of a sort. This is not material to the example so I am keeping it simple. Feel free to elaborate :-)

Now we need to set up the metadata which the B2B infrastructure needs to identify the trading partnership agreement to use for sending this message out to the remote partner.

Switch to the QRY2ADRMediator mplan. Click the "Assign Values" button.



Create value assignments as follows:

From	To
constant: ADR_A19_231	property: b2b.documentDefinitionName
property: jca.jms.JMSProperty.documentProtocolName	property: b2b.documentProtocolName
property: jca.jms.JMSProperty.documentProtocolVersion	property: b2b.documentProtocolVersion
constant: ADR	property: b2b.documentTypeName
property: jca.jms.JMSProperty.toTradingPartnerId	property: b2b.fromTradingPartnerId
expression: oraext:generate-guid()	property: b2b.messageId
constant: 2	property: b2b.messageType
property: jca.jms.JMSProperty.messageId	property: b2b.replyToMessageId
property: jca.jms.JMSProperty.fromTradingPartnerId	property: b2b.toTradingPartnerId

From	To
constant : ADR_A19_231	property : b2b.documentDefinitionName
property : jca.jms.JMSProperty.documentProtocolName	property : b2b.documentProtocolName
property : jca.jms.JMSProperty.documentProtocolVersion	property : b2b.documentProtocolVersion
constant : ADR	property : b2b.documentTypeName
property : jca.jms.JMSProperty.toTradingPartnerId	property : b2b.fromTradingPartnerId
expression : oraext:generate-guid()	property : b2b.messageId
constant : 2	property : b2b.messageType
property : jca.jms.JMSProperty.messageId	property : b2b.replyToMessageId
property : jca.jms.JMSProperty.fromTradingPartnerId	property : b2b.toTradingPartnerId

"Save All" and "Deploy" the project.

Test the Solution

Assume that the HL7 Sender client JAR is in c:\tools\CMDHL7.

Assume the test file, QRY_A19._ID_D224_3716691_HL7.2.3.1.~in, is in C:\hl7\2.3.1_A19.

Replace the JDK Path with the path to your JDK.

Open a command window and issue the following commands:

```
cd C:\tools\CMDHL7
C:\jdk1.6.0_20\bin\java.exe -Djava.util.logging.config.file=logging_info.properties
-jar CMDHL7Sender_v0.7.jar -h localhost -p 21100 -t 60000 -a CLI1 -b A19QRY -x GWYQ -y A19ADR
-c A -q -f c:\hl7\2.3.1_A19\QRY_A19._ID_D224_3716691_HL7.2.3.1.~in
```

The request will get logged, something along the lines of:

```
08/04/2011 2:57:34 PM au.id.czapski.hl7.CMDHL7Sender main
INFO: Sending Message:
MSH|^~\&|CLI1|A19QRY|GWYQ|A19ADR|200607031745||QRY^A19|A_000000|P|2.3.1
QRD|200607031745|R|I|Q1004|||1^RD|3716691^^^^^^^D224|DEM
```

The response will get logged along the lines of:

```
08/04/2011 2:57:43 PM au.id.czapski.hl7.CMDHL7Sender main
INFO: Received response:
MSH|^~\&|GWYQ|A19ADR|CLI1|A19QRY|200607031745||ADR^A19|ACK_A_000000|P|2.3.1
MSA|AA|A_000000
QRD|200607031745|R|I|Q1004|||1^RD|3716691
PID|||3716691||Smith^John||19960111|M|||||||0908070605
```

The A19 Query processor is working.

Summary

We configured the HL7 v2 delimited Request/Response infrastructure for processing A19 Query messages and returning ADR responses. The Oracle SOA Suite 11g R1 PS3 B2B HL7 infrastructure was used to take care of HL7 MLLP communication and HL7 v2 delimited transformation to/from XML. The SOA Suite Mediator-based

Composite was used to take care processing the QRY request and producing the ADR response.

The B2B Web Console Metrics page shows partner interaction.

Active Trading Partners		
Name	No. Of Messages Processed	
	From	To
A19QRY	1	1
B2BGateway	1	1

The B2B Web Console Metrics page shows documents exchanged.

Active Document Types		
Name	No. Of Messages Processed	
	Outbound	Inbound
HL7-2.3.1-ADR	1	0
HL7-2.3.1-QRY	0	1

The B2B Web Console Reports page shown messages exchanged, state, type, partners involved, etc..

Result (Message Count: 2) Sort by: Order

	Details	State	Document Type	Agreement	Sender	Receiver
1		MSG_COMPLETE	ADR	A19ADR_231_Out_Agr	B2BGateway	A19QRY
2		MSG_COMPLETE	QRY	A19QRY_231_In_Agr	A19QRY	B2BGateway

One can drill down into the payload of the wire message to see the actual HL7 delimited data.

Payload ✖

[Download](#)

```
MSH|^~\&|GWYQ^^|A19ADR^^|CLI1^^|A19QRY^^|200607031745^^||ADR^A19|ACK_A_000000|P
```