Impact2014

April 27 - May 1 Las Vegas, NV

3161A
Healthcare Integration Lab
Lab Instructions

Author:
Ben Thompson, Lead Architect IBM Integration Bus Industry Packs

1. Lab Objectives

In this lab, you will investigate the IBM Integration Bus Healthcare Pack's HL7 to HL7 DFDL Pattern, learn how to configure various pattern parameters, and see the Integration Bus message flows that are generated as a result of setting these parameters. You will deploy the resulting message flows and use the integrated testing tools provided with the Healthcare Pack to test the deployed applications. You will also try out the monitoring tools which are provided with the IBM Integration Bus Healthcare Pack

If you are using the pre-configured VMware image, the Integration Bus runtime configuration has already been created for you; this is the default node (from version 9, a "broker" is now named a "node"), named `IB9NODE` and the default queue manager named `IB9QMGR`. We will use a server (from version 9, an "execution group" is now named a "server") named `default`. The IBM Integration Bus Healthcare Pack (v3.0.0.0) has been installed, and the required post-installation tasks have already been carried out to extend the node to be able to run Healthcare Pack content. As a reminder, in case you are not using the pre-configured VMWare image, the post-installation tasks are shown below. These should be issued in an IBM Integration Console (you will find a shortcut on the Start menu to launch the Console).

If you are using the provided VMWare image you do not have to run these commands!

```
mqsistop BROKERNAME
mqsichangebroker BROKERNAME –f 9.0.0.1
mqsistart BROKERNAME
mqsimode BROKERNAME -x “healthcare,medicalDevices”
```

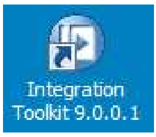
If you are using your own installation, please make sure the above commands are completed.

If you are using the provided VMWare image, log on using `wmbadmin` / `passw0rd`

2. HL7 to HL7 DFDL Pattern

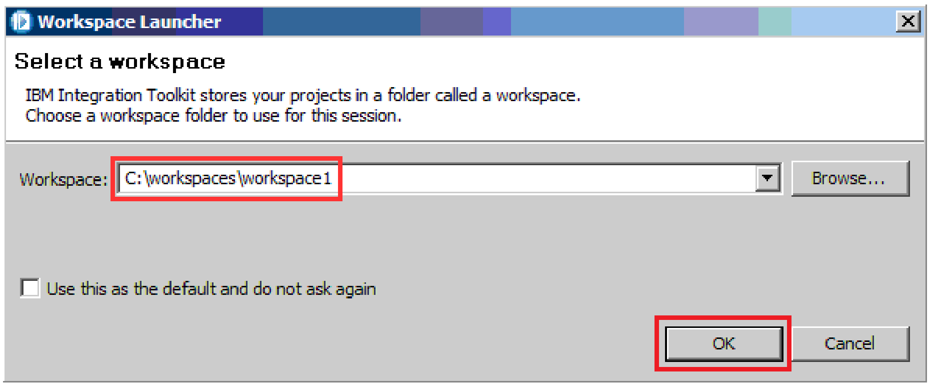
a. Creating the pattern Instance

1. Start the Integration Toolkit 9.0.0.1 using the shortcut on the Desktop:



.

At the workspace prompt, specify your own name for the workspace in the directory `C:\workspaces` and click `OK`. For example, type `C:\workspaces\workspace1`



Workspace Launcher

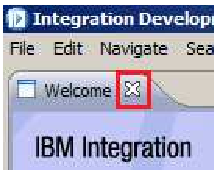
Select a workspace

IBM Integration Toolkit stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

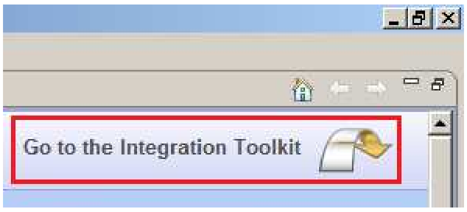
Workspace: C:\workspaces\workspace1

□ Use this as the default and do not ask again

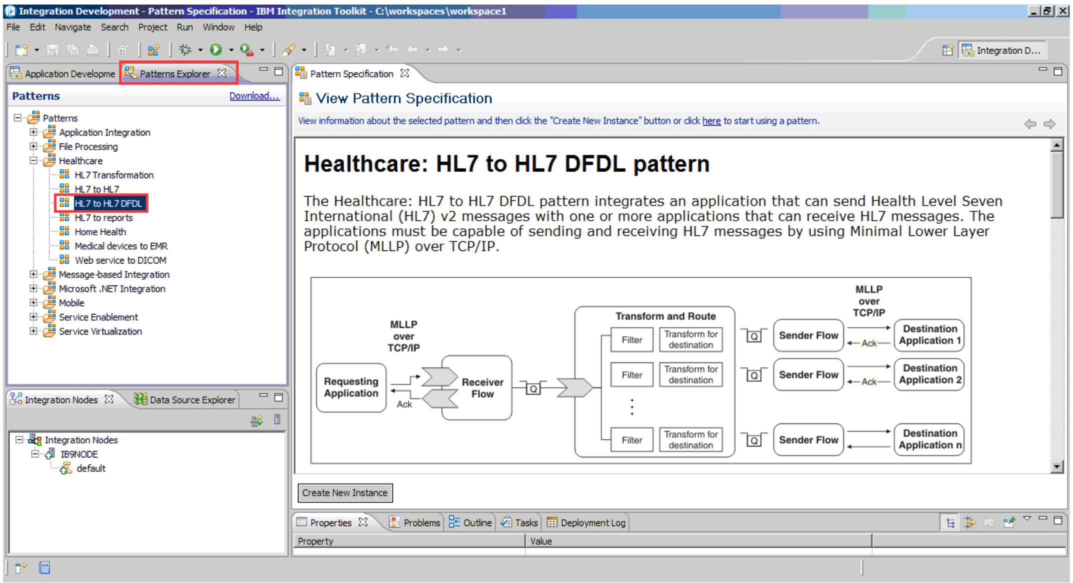
When the workspace has opened, close the Welcome pane by clicking on the cross symbol in the top left corner, or click on the link named “Go to the Integration Toolkit” in the top right corner to reveal the Integration Toolkit:



or



2. Select the Patterns Explorer, and then select the `HL7 to HL7 DFDL` pattern underneath the Healthcare category:



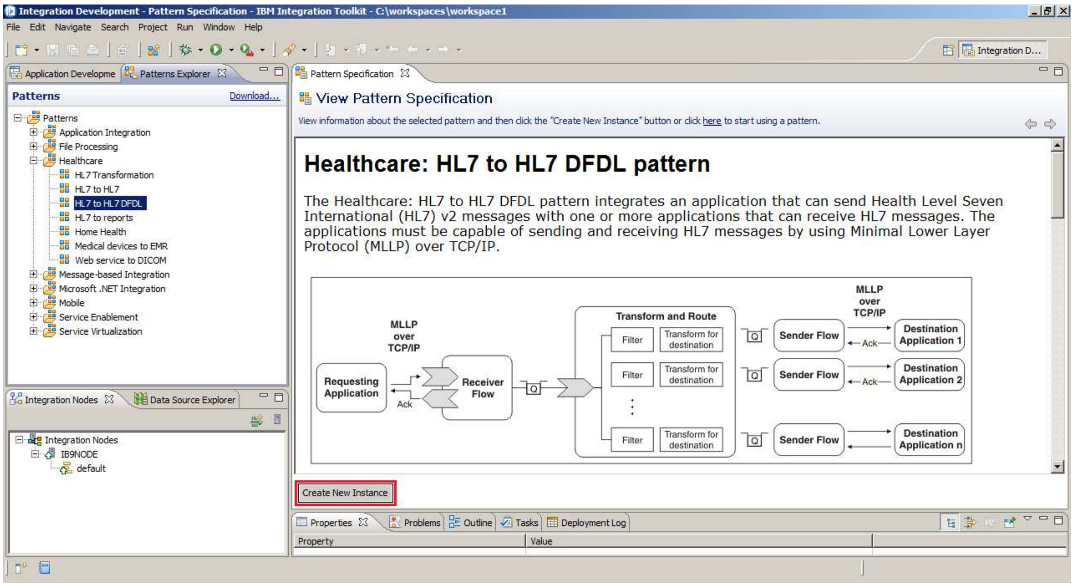
View information about the selected pattern and then click the "Create New Instance" button or click here to start using a pattern.

Healthcare: HL7 to HL7 DFDL pattern

The Healthcare: HL7 to HL7 DFDL pattern integrates an application that can send Health Level Seven International (HL7) v2 messages with one or more applications that can receive HL7 messages. The applications must be capable of sending and receiving HL7 messages by using Minimal Lower Layer Protocol (MLLP) over TCP/IP.

The pattern specification will open. The pattern is designed to receive input from a requesting application, which sends HL7 messages using the MLLP protocol. It can send output data to several target applications, known as destinations. Take some time to investigate the pattern specification.

3. When you have finished reading the specification, click the `Create New Instance` button:

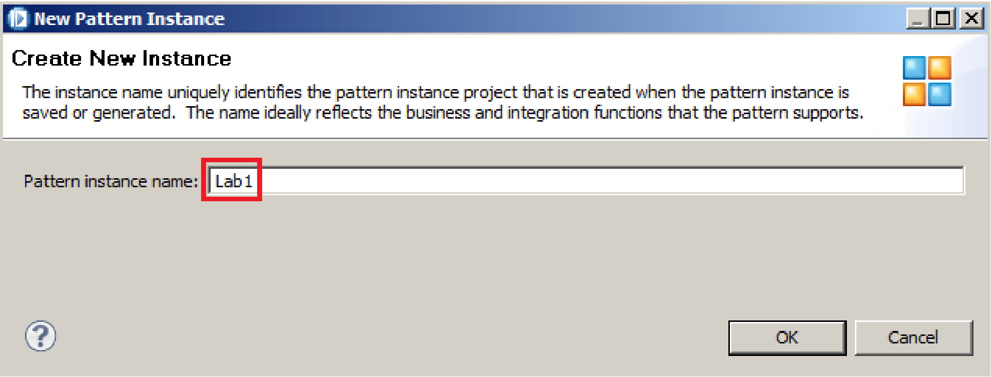


View information about the selected pattern and then click the "Create New Instance" button or click here to start using a pattern.

Healthcare: HL7 to HL7 DFDL pattern

The Healthcare: HL7 to HL7 DFDL pattern integrates an application that can send Health Level Seven International (HL7) v2 messages with one or more applications that can receive HL7 messages. The applications must be capable of sending and receiving HL7 messages by using Minimal Lower Layer Protocol (MLLP) over TCP/IP.

4. Specify a name for the new instance, for example, `Lab1`, and click OK:

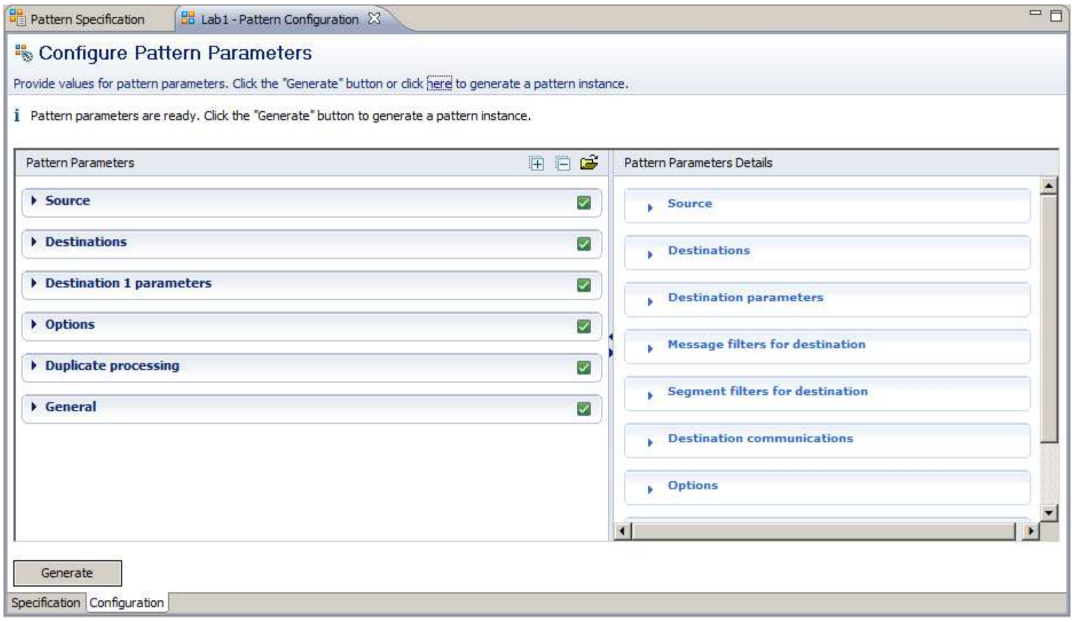


Create New Instance

The instance name uniquely identifies the pattern instance project that is created when the pattern instance is saved or generated. The name ideally reflects the business and integration functions that the pattern supports.

Pattern instance name: Lab1

5. After a few seconds, the Configure Pattern Parameters editor will open:



Configure Pattern Parameters

Provide values for pattern parameters. Click the "Generate" button or click here to generate a pattern instance.

Pattern parameters are ready. Click the "Generate" button to generate a pattern instance.

Pattern Parameters

- Source ☑
- Destinations ☑
- Destination 1 parameters ☑
- Options ☑
- Duplicate processing ☑
- General ☑

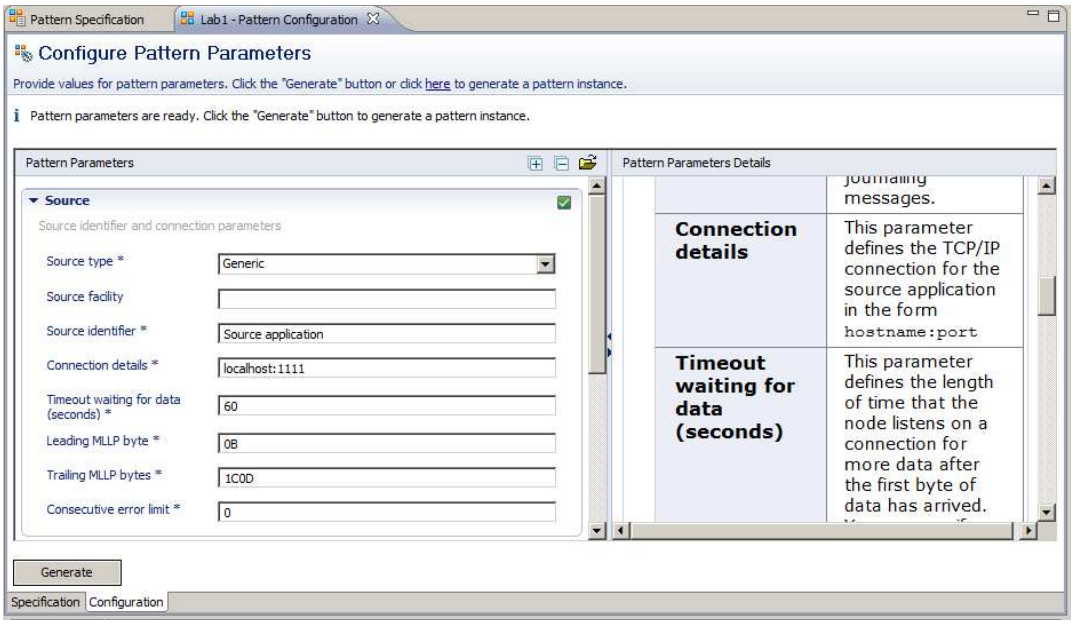
Pattern Parameters Details

- Source
- Destinations
- Destination parameters
- Message filters for destination
- Segment filters for destination
- Destination communications
- Options

6. Expand the `Source` group in the pattern parameters, and the `Source` group in the Details pane on the right.

   We won't make any changes to the parameters in this section, but take the time to look at the description of each parameter in the Details pane.

   Note that the `Connection details` for the Source system are set to a default value of `localhost:1111`. This corresponds to the source application in the set of Test Applications that we will end up using to drive this scenario.



Configure Pattern Parameters

Provide values for pattern parameters. Click the "Generate" button or click here to generate a pattern instance.

Pattern parameters are ready. Click the "Generate" button to generate a pattern instance.

Pattern Parameters

Source ☑

Source identifier and connection parameters

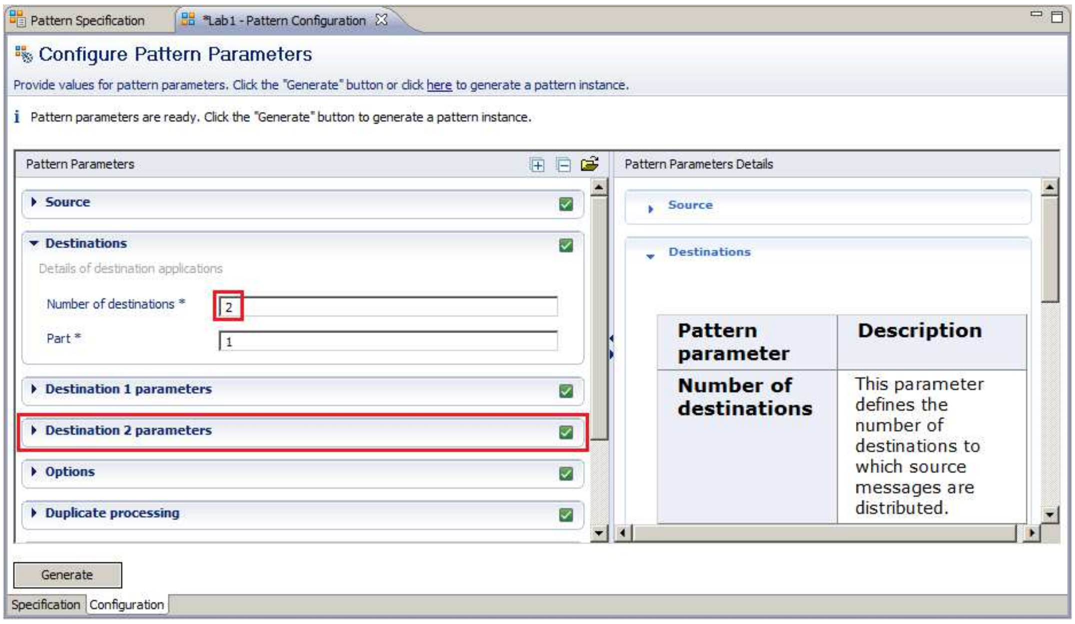
Source type *	Generic
Source facility	
Source identifier *	Source application
Connection details *	localhost:1111
Timeout waiting for data (seconds) *	60
Leading MLLP byte *	0B
Trailing MLLP bytes *	1C0D
Consecutive error limit *	0

Pattern Parameters Details

	journaling messages.
Connection details	This parameter defines the TCP/IP connection for the source application in the form `hostname:port`
Timeout waiting for data (seconds)	This parameter defines the length of time that the node listens on a connection for more data after the first byte of data has arrived.

7. Collapse the `Source` group in the pattern parameters, and expand the `Destinations` group.

   Set the `Number of destinations` property to be 2. This will automatically add a new group, `Destination2 parameters`.



Configure Pattern Parameters

Provide values for pattern parameters. Click the "Generate" button or click here to generate a pattern instance.

Pattern parameters are ready. Click the "Generate" button to generate a pattern instance.

Pattern Parameters

Source ☑

Destinations ☑

Details of destination applications

Number of destinations *	2
Part *	1

Destination 1 parameters ☑

Destination 2 parameters ☑

Options ☑

Duplicate processing ☑

Pattern Parameters Details

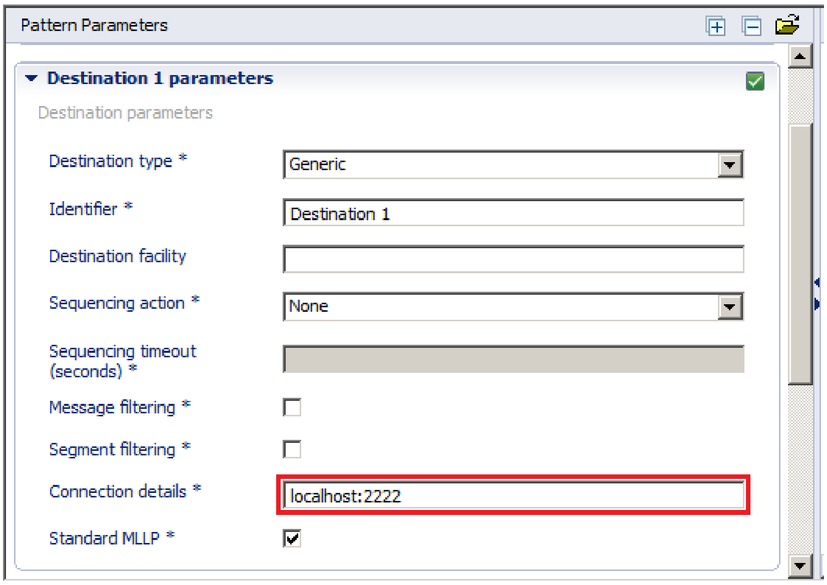
Source

Destinations

Pattern parameter	Description
Number of destinations	This parameter defines the number of destinations to which source messages are distributed.

8. Expand the `Destination1` Parameters group.

   Take a look at the details for each of these parameters.



Pattern Parameters

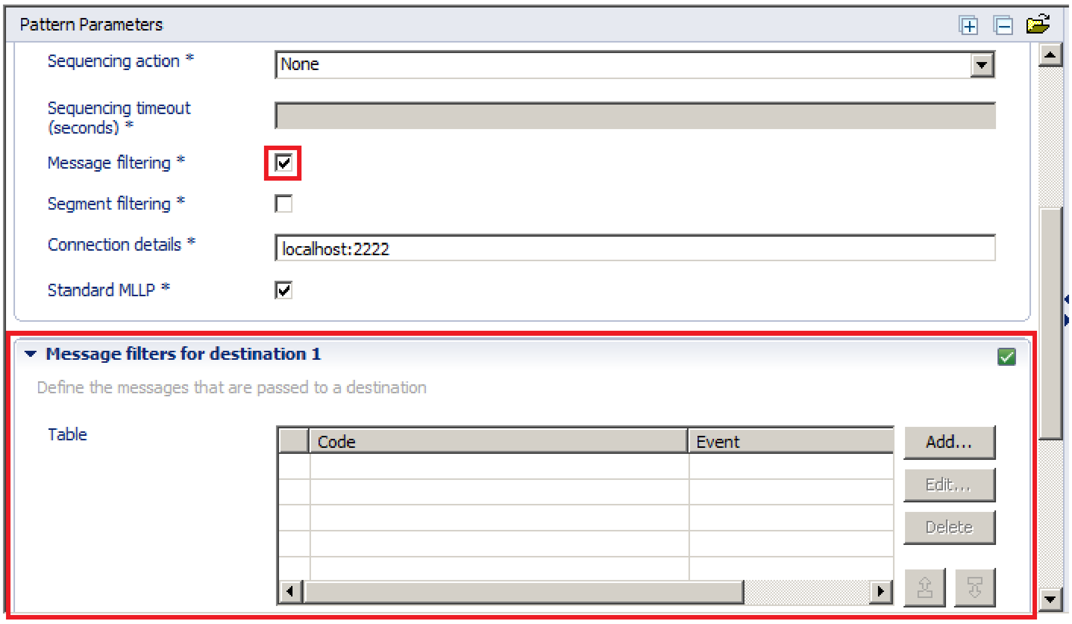
Destination 1 parameters

Destination parameters

Destination type *	Generic
Identifier *	Destination 1
Destination facility	
Sequencing action *	None
Sequencing timeout (seconds) *	
Message filtering *	□
Segment filtering *	□
Connection details *	localhost:2222
Standard MLLP *	☑

   Note that the connection details are set to the default of `localhost:2222`. As with the source, this corresponds to the Test Applications that we will be using later.

9. We won't make any permanent changes at this stage. However, we will investigate a couple of parameters. Select the `Message filtering` checkbox. Note that the pattern configuration will dynamically change. A `Message filters for destination 1` group is added to the display. Expand this group:



Pattern Parameters

Sequencing action *	None
Sequencing timeout (seconds) *	
Message filtering *	☑
Segment filtering *	□
Connection details *	localhost:2222
Standard MLLP *	☑

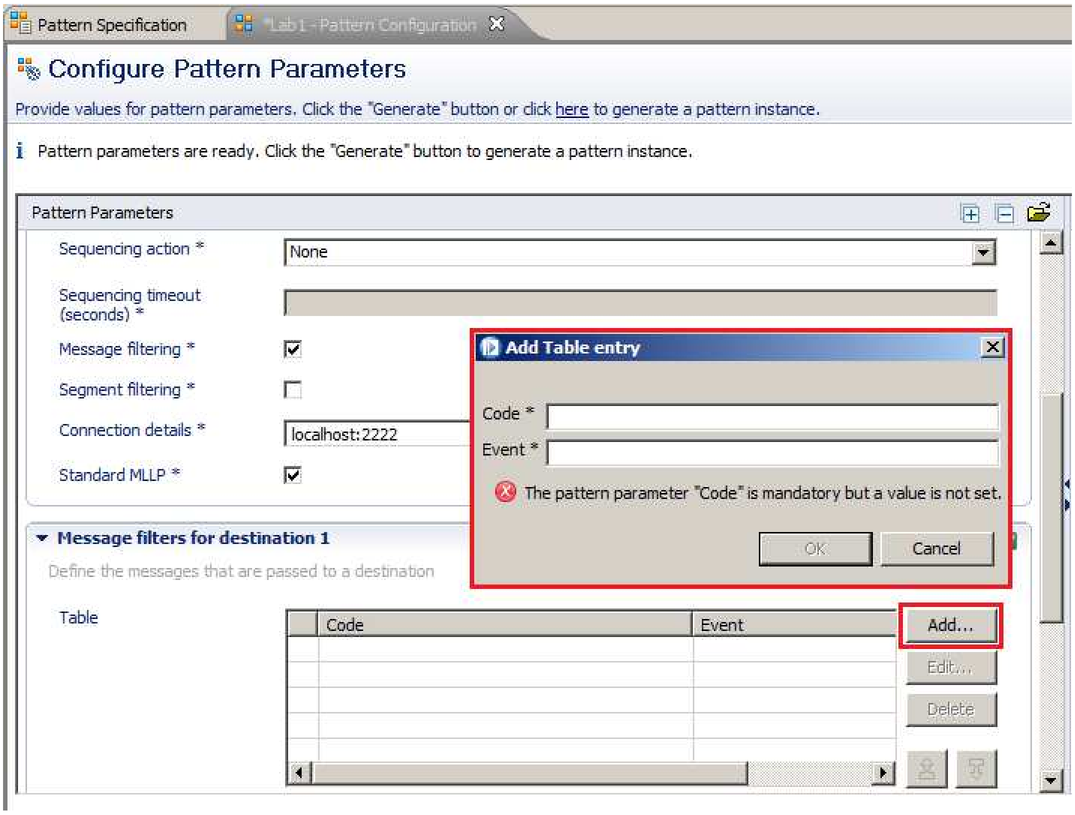
Message filters for destination 1

Define the messages that are passed to a destination

Table

Code	Event

10. Click the Add button to add a new message filter, and you will see a dialog presented which enables you to specify a filter which dictates which messages are passed to a particular destination.



Configure Pattern Parameters

Provide values for pattern parameters. Click the "Generate" button or click here to generate a pattern instance.

Pattern parameters are ready. Click the "Generate" button to generate a pattern instance.

Pattern Parameters

Sequencing action * None

Sequencing timeout (seconds) *

Message filtering * ☑

Segment filtering * □

Connection details * localhost:2222

Standard MLLP * ☑

Message filters for destination 1

Define the messages that are passed to a destination

Table

Code	Event

Add Table entry

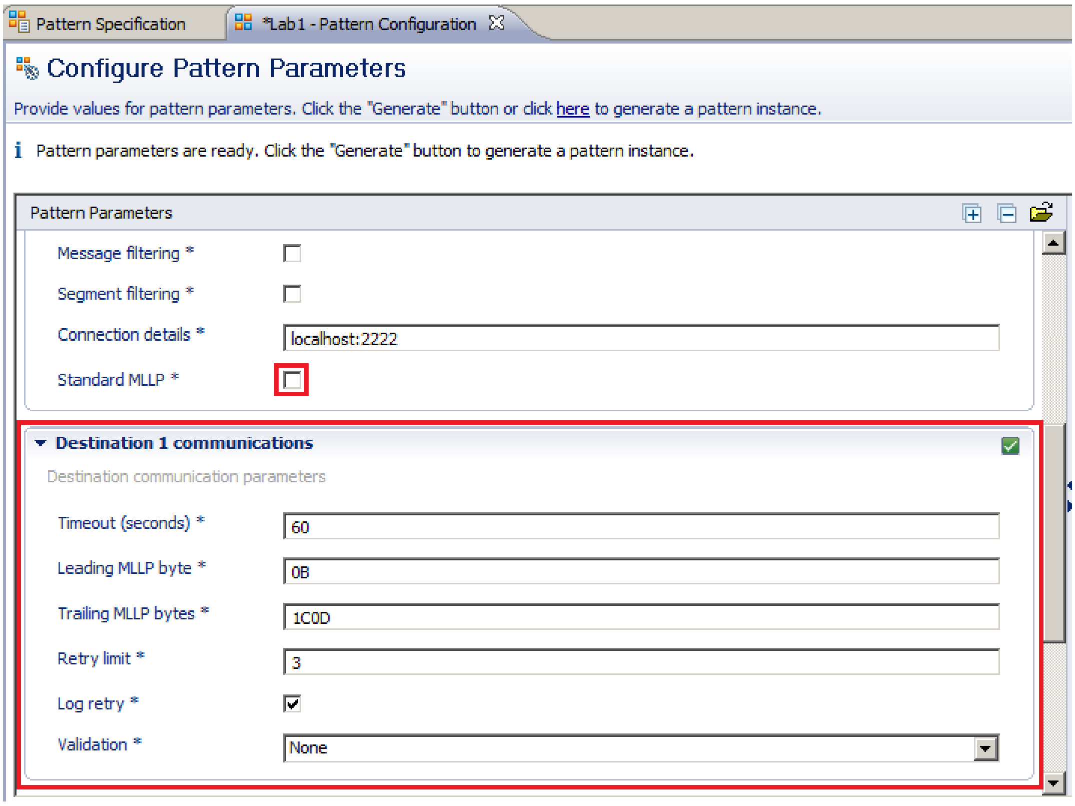
Code *

Event *

The pattern parameter "Code" is mandatory but a value is not set.

We won’t take this particular configuration any further, so just click `Cancel`, and then untick the `Message filtering` checkbox again. The `Message filters for destination 1` group will disappear. If you have time left at the end of this lab you might want to investigate the Message Filters function further yourself.

11. Next, deselect the `Standard MLLP` checkbox. This will dynamically add the `Destination 1 communications` group. This is where you can specify your own parameters for the target destination. We will not do this at the moment, so place a tick back in the `Standard MLLP` checkbox.



Configure Pattern Parameters

Provide values for pattern parameters. Click the "Generate" button or click here to generate a pattern instance.

Pattern parameters are ready. Click the "Generate" button to generate a pattern instance.

Pattern Parameters

Message filtering * □

Segment filtering * □

Connection details * localhost:2222

Standard MLLP * □

Destination 1 communications ☑

Destination communication parameters

Timeout (seconds) * 60

Leading MLLP byte * 0B

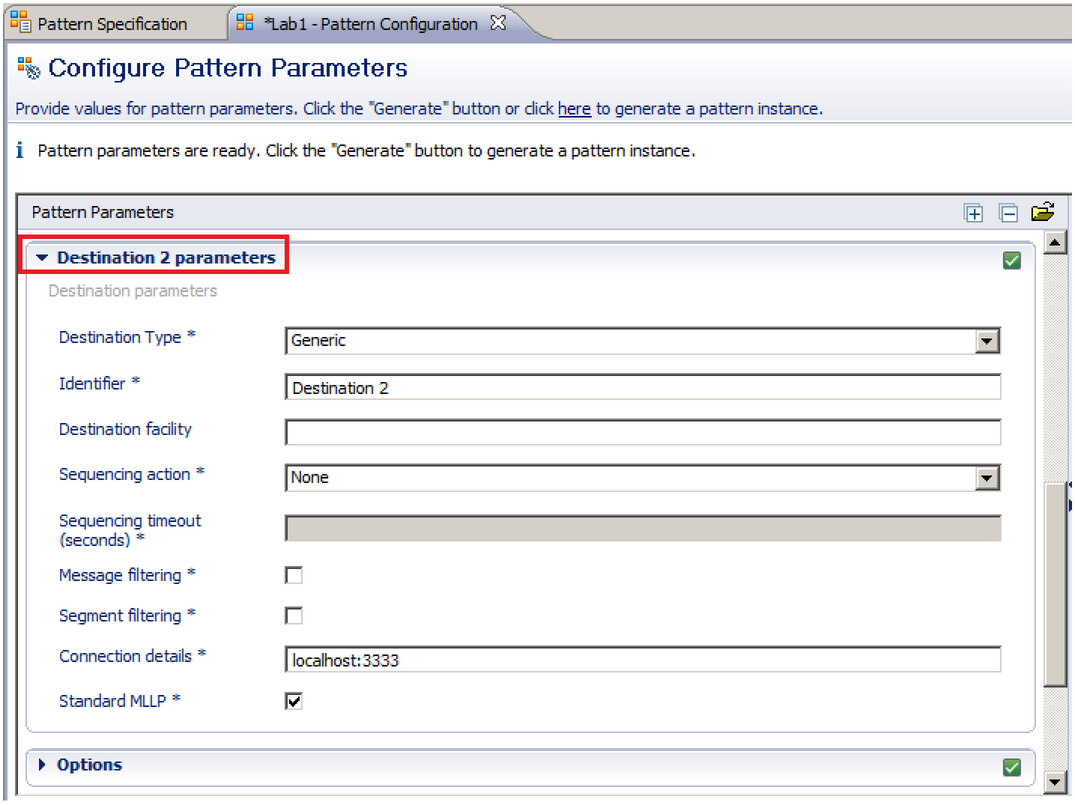
Trailing MLLP bytes * 1C0D

Retry limit * 3

Log retry * ☑

Validation * None

12. Collapse the `Destination 1` Parameters group. Expand the `Destination 2` Parameters group. Observe that each destination contains the same set of parameters. Do not make any changes.



Configure Pattern Parameters

Provide values for pattern parameters. Click the "Generate" button or click here to generate a pattern instance.

Pattern parameters are ready. Click the "Generate" button to generate a pattern instance.

Pattern Parameters

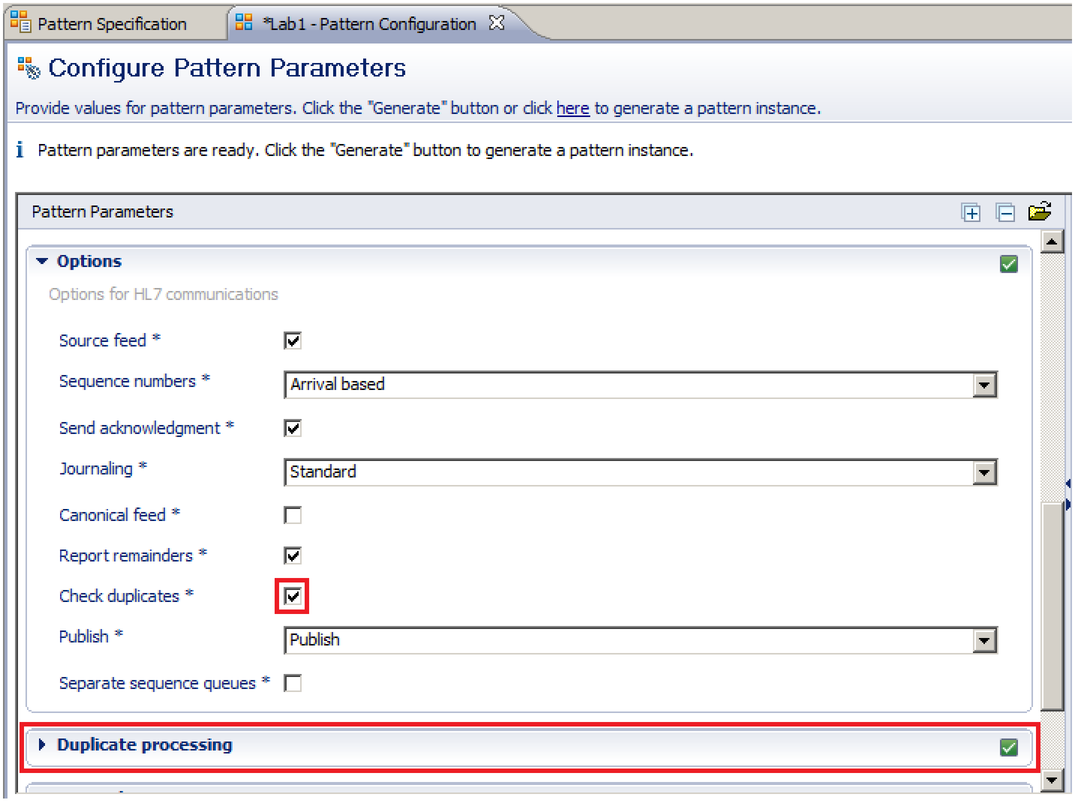
Destination 2 parameters

Destination parameters

Destination Type *	Generic
Identifier *	Destination 2
Destination facility	
Sequencing action *	None
Sequencing timeout (seconds) *	
Message filtering *	□
Segment filtering *	□
Connection details *	localhost:3333
Standard MLLP *	☑

Options

13. Expand the `Options` group. Look at the corresponding specifications for this group. Note that the `Check duplicates` parameter is checked, and that there is a group called `Duplicate processing`:



Configure Pattern Parameters

Provide values for pattern parameters. Click the "Generate" button or click here to generate a pattern instance.

i Pattern parameters are ready. Click the "Generate" button to generate a pattern instance.

Pattern Parameters

▾ Options ☑

Options for HL7 communications

Source feed * ☑

Sequence numbers * Arrival based

Send acknowledgment * ☑

Journaling * Standard

Canonical feed * □

Report remainders * ☑

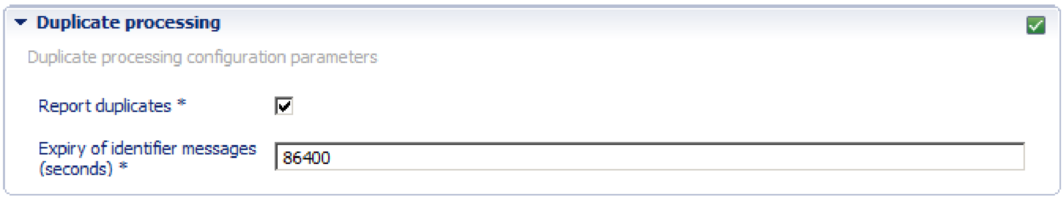
Check duplicates * ☑

Publish * Publish

Separate sequence queues * □

▸ Duplicate processing ☑

14. Expand the `Duplicate processing` group.



▾ Duplicate processing ☑

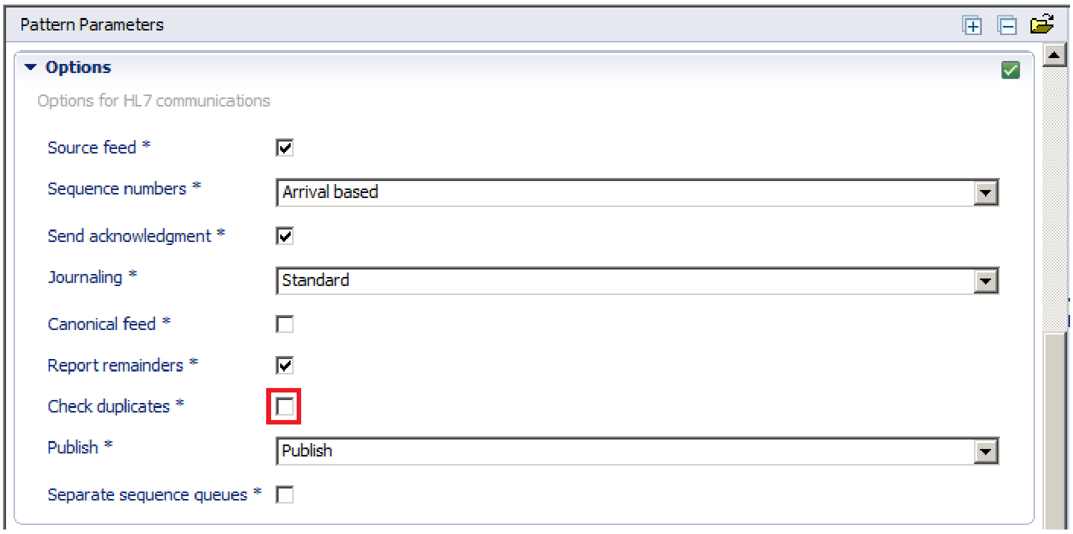
Duplicate processing configuration parameters

Report duplicates * ☑

Expiry of identifier messages (seconds) * 86400

These pattern parameters control whether the generated flows should execute duplicate processing to check whether inbound messages are duplicates of previously received messages, and reported as duplicates.

15. However, in this example, we will remove the duplicate checking. This will make the test cases easier to run, and the tests won't fail as a result of duplicate checking failure. Uncheck the `Check duplicates` checkbox. The `Duplicate processing` group will disappear.



Pattern Parameters

Options

Options for HL7 communications

Source feed * ☑

Sequence numbers * Arrival based

Send acknowledgment * ☑

Journaling * Standard

Canonical feed * □

Report remainders * ☑

Check duplicates * □

Publish * Publish

Separate sequence queues * □

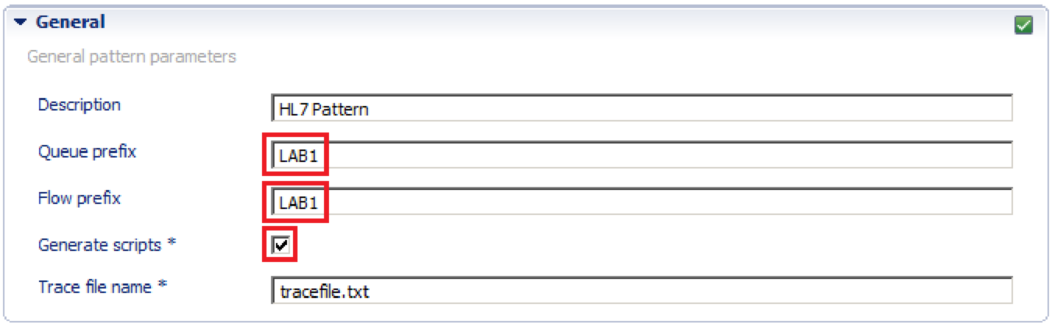
Collapse the Options Group.

16. Expand the `General` group, and make the following changes:

Set `Queue prefix = LAB1` (or a prefix of your choice). This is used to specify queue names in the generated message flows; these queues are also included in the MQSC script that is created by the pattern.

Set `Flow prefix = LAB1` (or a prefix of your choice). This is used to uniquely define the generated flow names.

Select the `Generate scripts` checkbox. This means that the pattern will create an MQSC script which can be used to create the MQ queues.



General

General pattern parameters

Description HL7 Pattern

Queue prefix LAB1

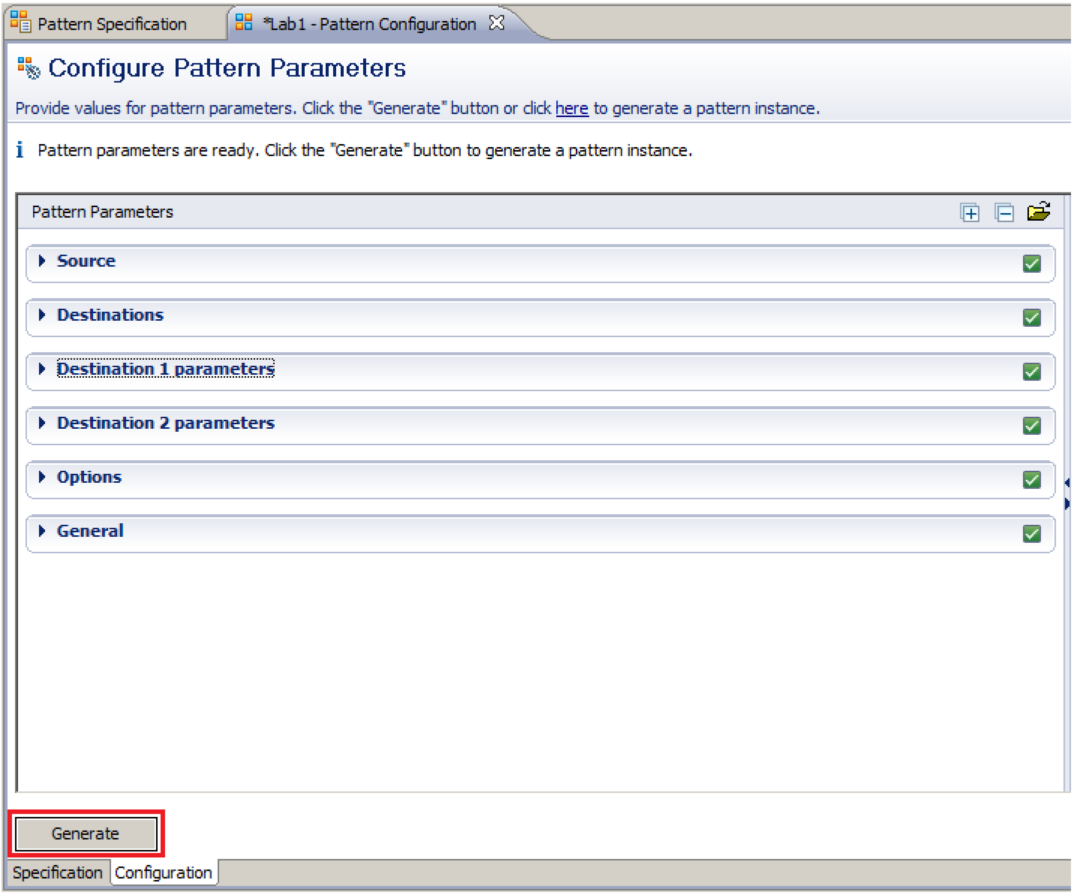
Flow prefix LAB1

Generate scripts * ☑

Trace file name * tracefile.txt

Note that the trace file name does not specify a directory. On Windows, the default location for this trace file is `C:\Program Files\ibm\MQSI\9.0.0.1\bin`.

17. You will see that all the required parameters have now been set (all the icons are green and ticked). You can now generate the application. Click the `Generate` button:



Configure Pattern Parameters

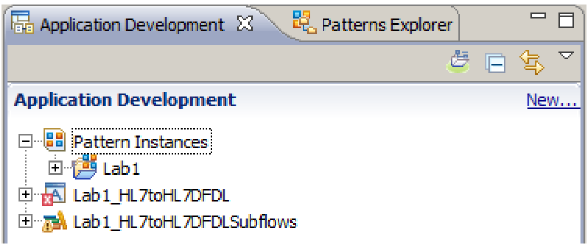
Provide values for pattern parameters. Click the "Generate" button or click here to generate a pattern instance.

Pattern parameters are ready. Click the "Generate" button to generate a pattern instance.

Pattern Parameters

- Source ☑
- Destinations ☑
- Destination 1 parameters ☑
- Destination 2 parameters ☑
- Options ☑
- General ☑

18. Since the pattern is quite large, the generate process may take a few minutes. Several progress windows will pop up during this process. When complete, you should find you have been returned to the Application Development tab. You will see that three new projects will have been created, one of which will have a red cross against it (**don't worry – this is perfectly normal and is explained in the next step below!**):



If you have followed the suggested naming conventions in this guide, you will have a Pattern Instance project named `Lab1`. The pattern instance project holds a record of the parameters you set for this pattern instance, and can be used to regenerate the flows if you need to. We won't be using this from now on, so you can close this project if you wish. The pattern will also have generated an Application named `Lab1_HL7toHL7DFDL` and a Library named `Lab1_HL7toHL7DFDLSubflows`.

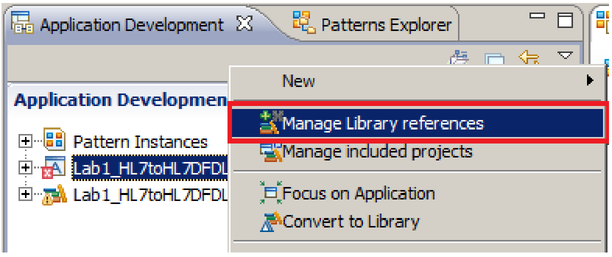
IIB applications and libraries are deployable containers of resources, such as message flows, subflows, message definitions (DFDL, XSD files), JAR files etc. An application is a container for all the resources that are required to create a solution. Applications typically contain message flows. A library is a logical grouping of related code, data, or both. A library typically contains reusable helper routines and resources such as subflows, ESQL modules, message definitions, maps, and Java utilities. Libraries can be referenced by applications, services, Message Broker projects, or other libraries.

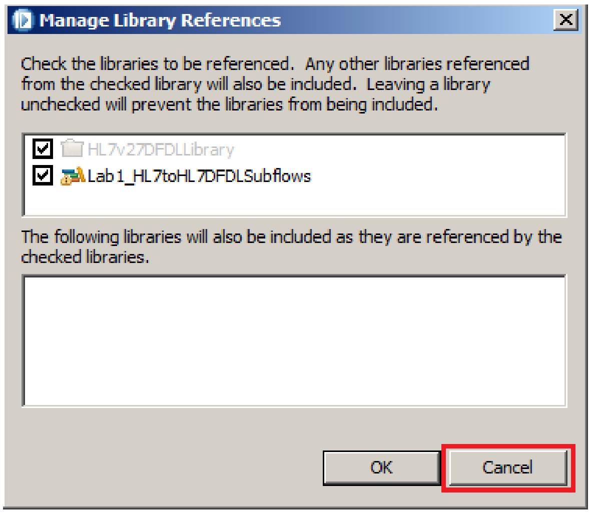
We will return to explore and test the generated message flows in a later section.

3. The HL7 Message Model

a. Importing the HL7 Message Model

1. `Lab1_HL7toHL7DFDL` requires an HL7 message model. You can see this dependency if you right click the project and select Manage Library references:




Manage Library References

Check the libraries to be referenced. Any other libraries referenced from the checked library will also be included. Leaving a library unchecked will prevent the libraries from being included.

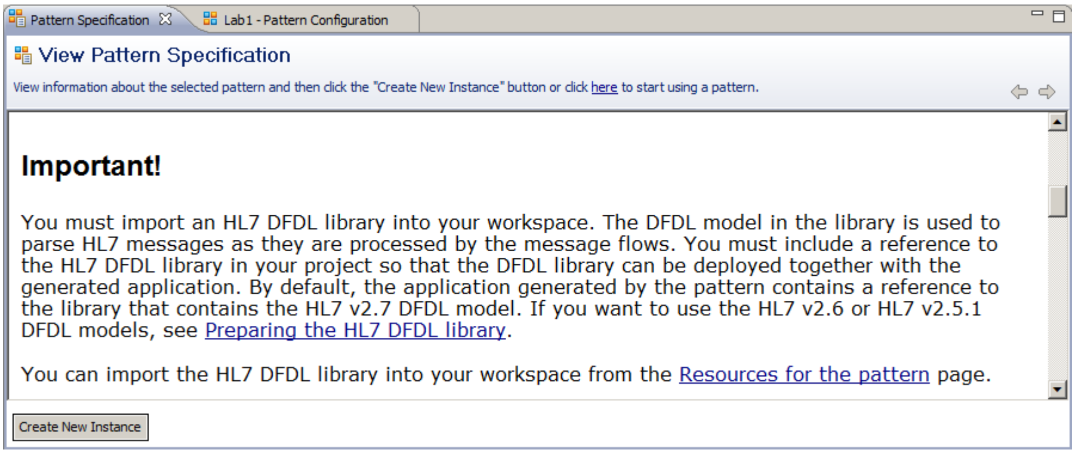
☑ HL7v27DFDLLibrary
☑ Lab1_HL7toHL7DFDLSubflows

The following libraries will also be included as they are referenced by the checked libraries.

Close the dialog by clicking the cancel button.

We have not yet imported the `HL7v27DFDLLibrary`, which is why the red cross is displayed against `Lab1_HL7toHL7DFDL`. To resolve this problem, we need to import the `HL7v27DFDLLibrary`, which we'll do next.

2. The Pattern Specification for the `HL7 to HL7 DFDL` pattern should still be open (if you have accidentally closed it, you can navigate back to it by switching to the Patterns Explorer tab). Scroll down and you will find a section, as shown below, entitled **Important!** Click the link for `Resources for the pattern`:



View Pattern Specification

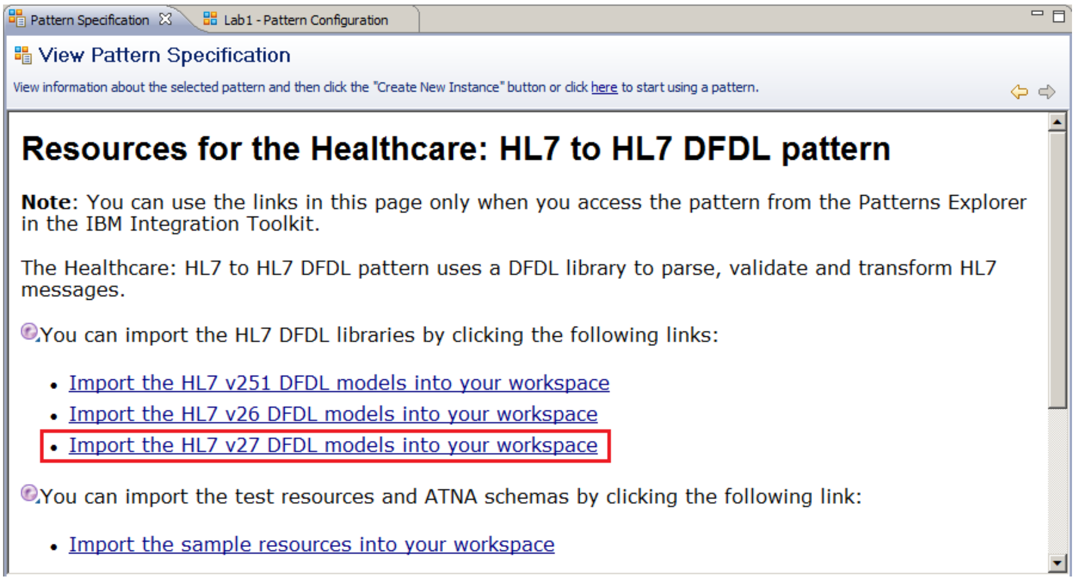
View information about the selected pattern and then click the "Create New Instance" button or click here to start using a pattern.

Important!

You must import an HL7 DFDL library into your workspace. The DFDL model in the library is used to parse HL7 messages as they are processed by the message flows. You must include a reference to the HL7 DFDL library in your project so that the DFDL library can be deployed together with the generated application. By default, the application generated by the pattern contains a reference to the library that contains the HL7 v2.7 DFDL model. If you want to use the HL7 v2.6 or HL7 v2.5.1 DFDL models, see Preparing the HL7 DFDL library.

You can import the HL7 DFDL library into your workspace from the Resources for the pattern page.

On the Resources page, click the link to `Import the HL7 v27 DFDL models into your workspace`:



View Pattern Specification

View information about the selected pattern and then click the "Create New Instance" button or click here to start using a pattern.

Resources for the Healthcare: HL7 to HL7 DFDL pattern

Note: You can use the links in this page only when you access the pattern from the Patterns Explorer in the IBM Integration Toolkit.

The Healthcare: HL7 to HL7 DFDL pattern uses a DFDL library to parse, validate and transform HL7 messages.

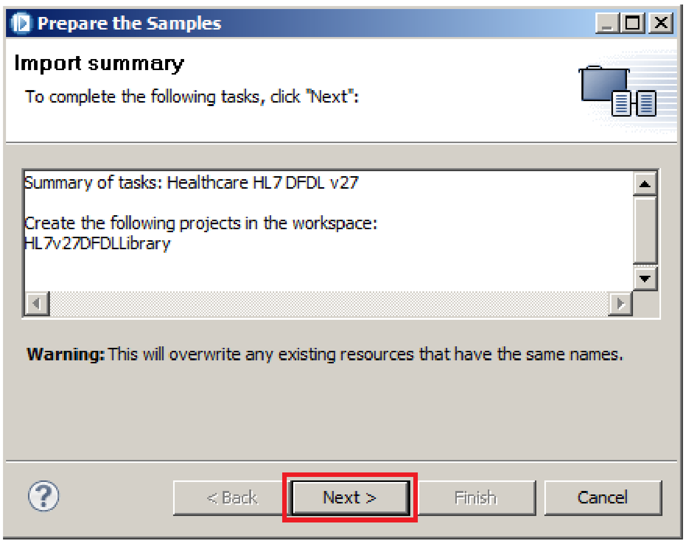
You can import the HL7 DFDL libraries by clicking the following links:

- Import the HL7 v251 DFDL models into your workspace
- Import the HL7 v26 DFDL models into your workspace
- Import the HL7 v27 DFDL models into your workspace

You can import the test resources and ATNA schemas by clicking the following link:

- Import the sample resources into your workspace

3. On the resulting dialog, click the `Next` button:



Prepare the Samples

Import summary

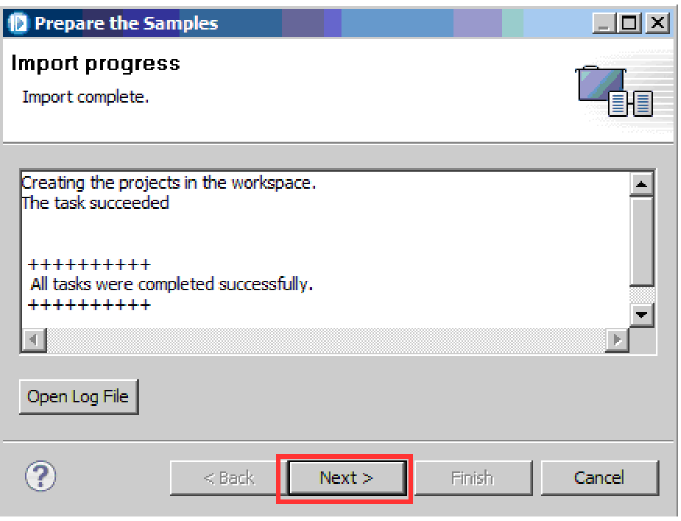
To complete the following tasks, click "Next":

```
Summary of tasks: Healthcare HL7 DFDL v27

Create the following projects in the workspace:
HL7v27DFDLLibrary
```

Warning: This will overwrite any existing resources that have the same names.

The import will complete, and you should click `Next` again:



Prepare the Samples

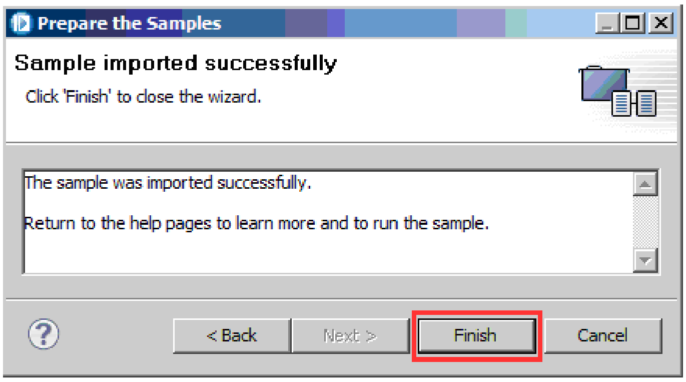
Import progress

Import complete.

```
Creating the projects in the workspace.
The task succeeded


 ++++++++++
 All tasks were completed successfully.
 ++++++++++
```

A success message is displayed:



Prepare the Samples

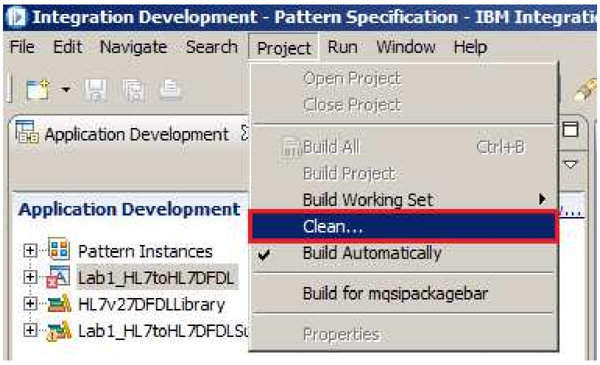
Sample imported successfully

Click 'Finish' to close the wizard.

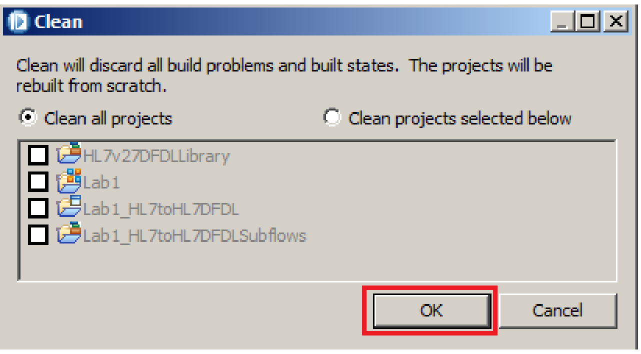
The sample was imported successfully.

Return to the help pages to learn more and to run the sample.

4. Select the project `Lab1_HL7toHL7DFDL` and from the `Project` menu, choose the option to `Clean`.



On the resulting dialog, click OK:

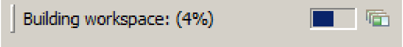


Clean will discard all build problems and built states. The projects will be rebuilt from scratch.

● Clean all projects ○ Clean projects selected below

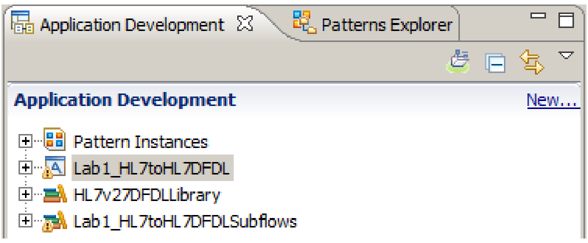
□ HL7v27DFDLLibrary
□ Lab1
□ Lab1_HL7toHL7DFDL
□ Lab1_HL7toHL7DFDLSubflows

The clean process can take a minute or two. At the bottom right of the Toolkit, you will see a progress message:

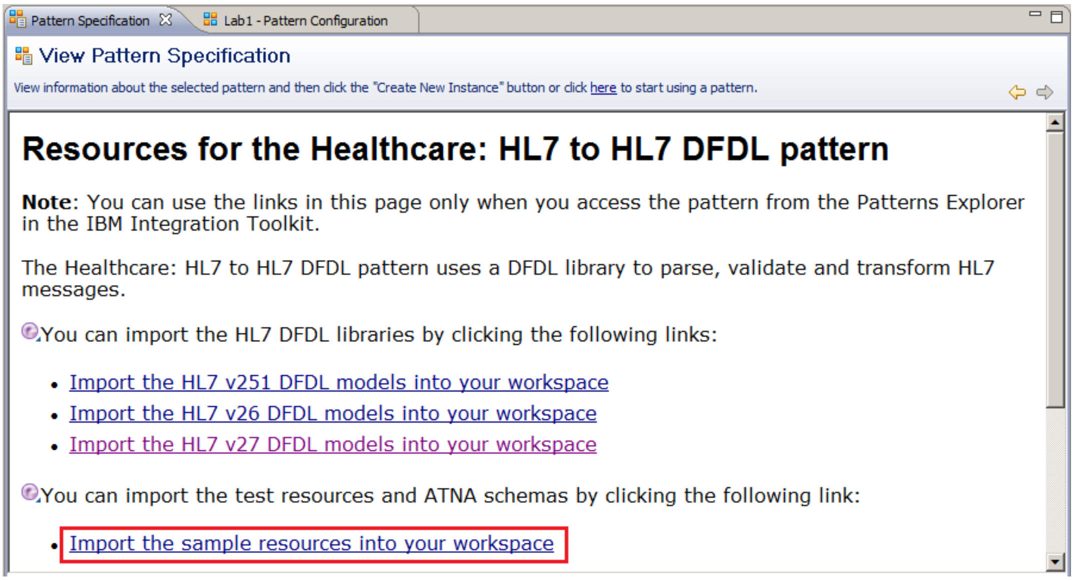


Building workspace: (4%)

When the clean process is complete, the project should no longer show any errors:



5. Now return to the Resources page, and select the link to `Import the sample resources into your workspace`:



View Pattern Specification

View information about the selected pattern and then click the "Create New Instance" button or click here to start using a pattern.

Resources for the Healthcare: HL7 to HL7 DFDL pattern

Note: You can use the links in this page only when you access the pattern from the Patterns Explorer in the IBM Integration Toolkit.

The Healthcare: HL7 to HL7 DFDL pattern uses a DFDL library to parse, validate and transform HL7 messages.

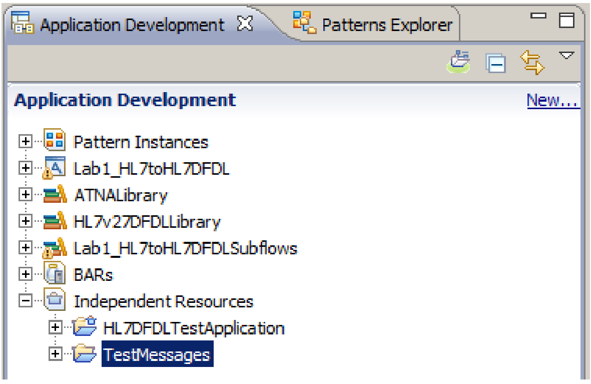
You can import the HL7 DFDL libraries by clicking the following links:

- Import the HL7 v251 DFDL models into your workspace
- Import the HL7 v26 DFDL models into your workspace
- Import the HL7 v27 DFDL models into your workspace

You can import the test resources and ATNA schemas by clicking the following link:

- Import the sample resources into your workspace

Click through the resulting dialog using the Next button, just like before. The dialog will import several other projects into your workspace, one of which contains test messages which we'll use to test the scenario later:

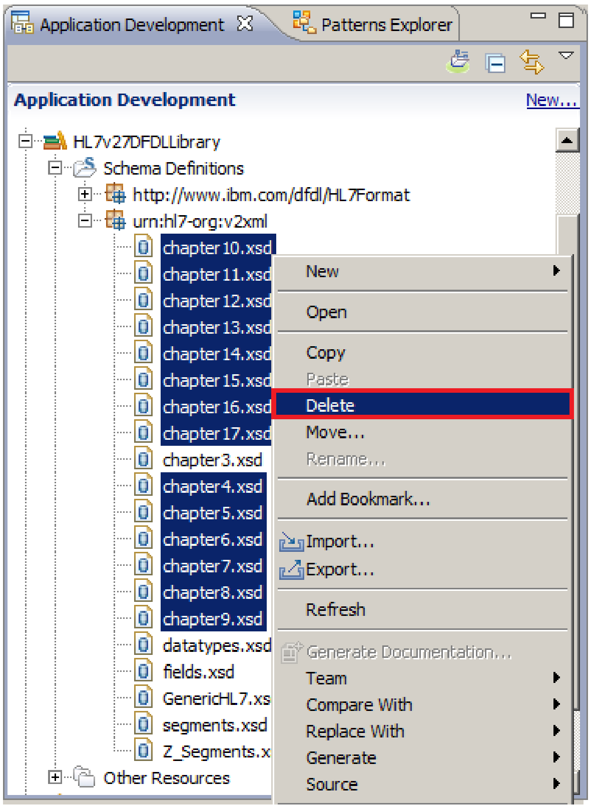


- Pattern Instances
- Lab1_HL7toHL7DFDL
- ATNALibrary
- HL7v27DFDLLibrary
- Lab1_HL7toHL7DFDLSubflows
- BARs
- Independent Resources
  - HL7DFDLTestApplication
  - TestMessages

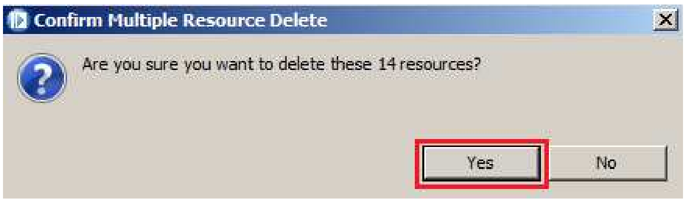
b.

c. Exploring the HL7 Message Model

1. In the Application Development view, expand the `HL7v27DFDLLibrary` in order to inspect the schemas it contains. As shown below, the library is quite large, and contains models for all the messages defined in the various chapters of the HL7 specification. We will only be using messages from chapter3, so to reduce the size of the model, highlight the other chapter schemas, using the CTRL key to do a multi-select, and then right click and choose `delete`:

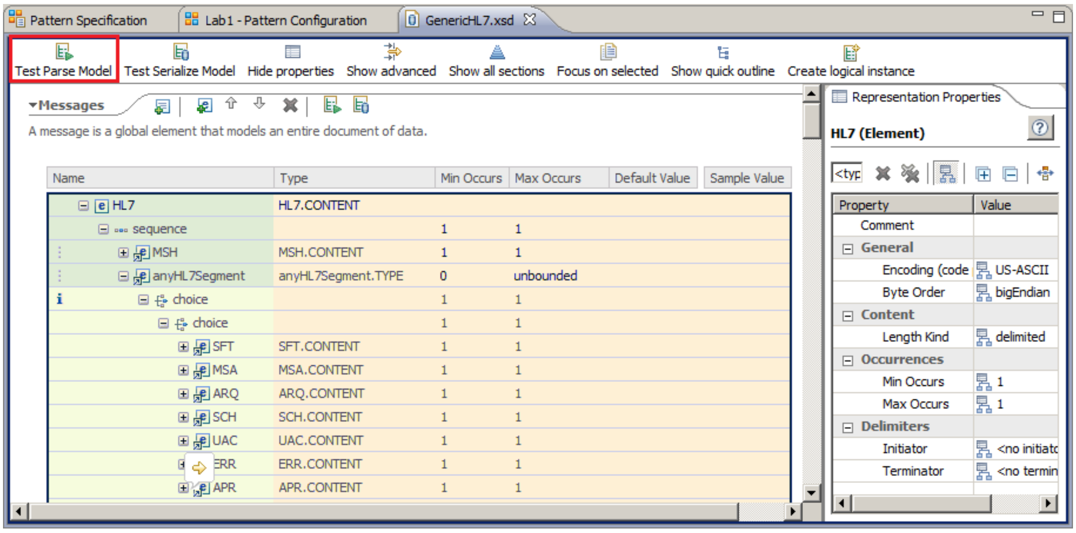


Click Yes on the warning pop-up:



Are you sure you want to delete these 14 resources?

2. When the delete completes, double click GenericHL7.xsd in the Application Development view (on the left) in order to open the schema with the DFDL editor:



A message is a global element that models an entire document of data.

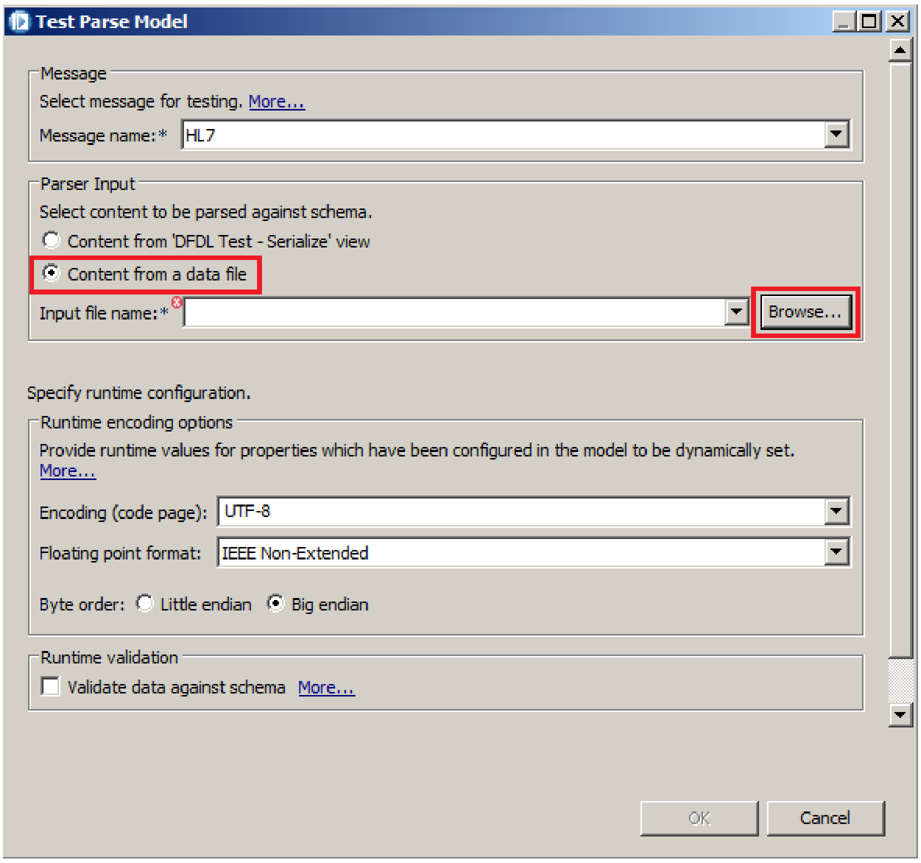
Name	Type	Min Occurs	Max Occurs	Default Value	Sample Value
HL7	HL7.CONTENT				
sequence		1	1		
MSH	MSH.CONTENT	1	1		
anyHL7Segment	anyHL7Segment.TYPE	0	unbounded		
choice		1	1		
choice		1	1		
SFT	SFT.CONTENT	1	1		
MSA	MSA.CONTENT	1	1		
ARQ	ARQ.CONTENT	1	1		
SCH	SCH.CONTENT	1	1		
UAC	UAC.CONTENT	1	1		
ERR	ERR.CONTENT	1	1		
APR	APR.CONTENT	1	1		

Property	Value
Comment	
General	
Encoding (code	US-ASCII
Byte Order	bigEndian
Content	
Length Kind	delimited
Occurrences	
Min Occurs	1
Max Occurs	1
Delimiters	
Initiator	<no initiatc
Terminator	<no termin

This schema contains a generic model which can be used to parse any HL7 message. As shown above, a choice structure is used in order that the model can accept input data which contains any set of HL7 segments, in any order after the MSH segment.

Click the `Test Parse Model` button.

3. The DFDL model can be tested in the Integration Toolkit without the need to deploy to the runtime. This is a very useful feature which speeds message flow development time. To specify the location of a test message, select the radio button named `Content from a data file` and then click the `Browse` button:



Test Parse Model

Message

Select message for testing. More...

Message name:* HL7

Parser Input

Select content to be parsed against schema.

○ Content from 'DFDL Test - Serialize' view

● Content from a data file

Input file name:*

Specify runtime configuration.

Runtime encoding options

Provide runtime values for properties which have been configured in the model to be dynamically set. More...

Encoding (code page): UTF-8

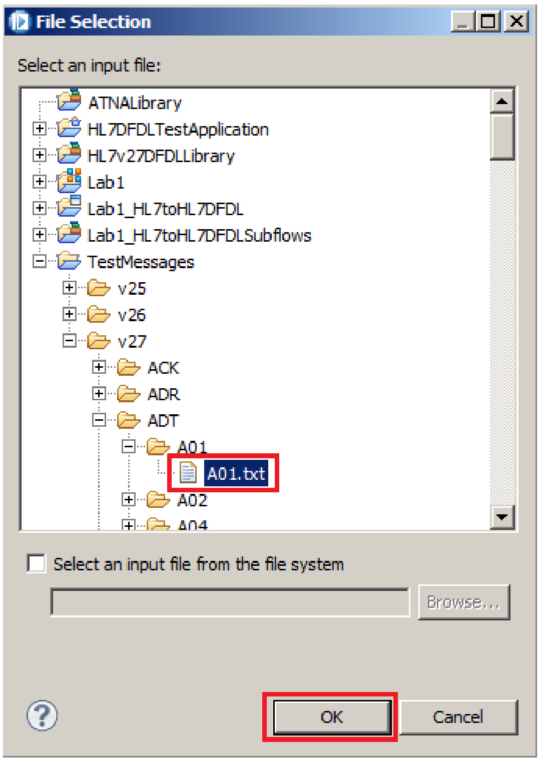
Floating point format: IEEE Non-Extended

Byte order: ○ Little endian ● Big endian

Runtime validation

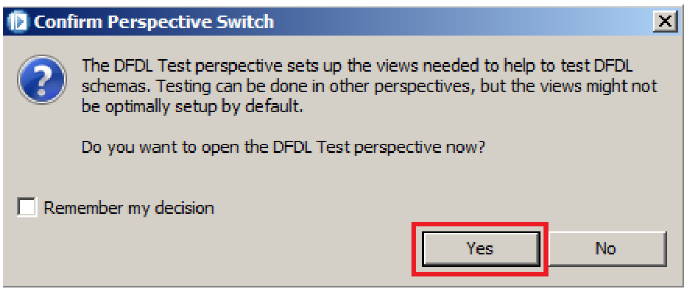
□ Validate data against schema More...

4. Navigate down the hierarchy and select `A01.txt` (an ADT_A01 message), and click OK:



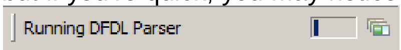
You will be returned to the previous window, where you should also click `OK`.

5. You will receive a warning that you are about to be placed into the DFDL Test perspective of the Toolkit. Click `Yes`.



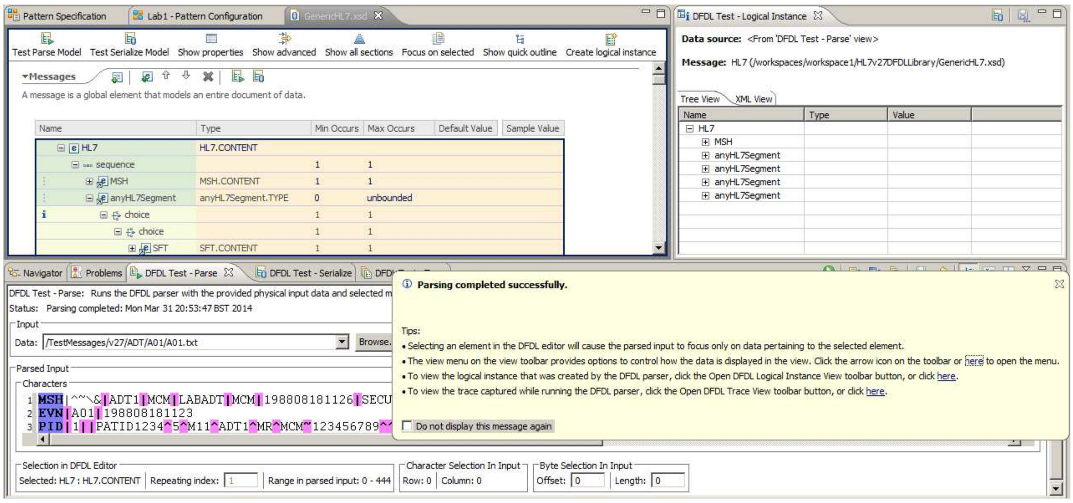
The DFDL Test perspective sets up the views needed to help to test DFDL schemas. Testing can be done in other perspectives, but the views might not be optimally setup by default.

Do you want to open the DFDL Test perspective now?

6. When the new set of views open, the DFDL Test parser will be invoked and will attempt to parse the selected test message using the generic HL7 model. The parse should only take a few seconds, but if you're quick, you may notice a status message in the bottom right corner of the toolkit: 

Running DFDL Parser

7. The result of the parse is shown (which should be successful!):



A message is a global element that models an entire document of data.

Name	Type	Min Occurs	Max Occurs	Default Value	Sample Value
HL7	HL7.CONTENT				
sequence		1	1		
MSH	MSH.CONTENT	1	1		
anyHL7Segment	anyHL7Segment.TYPE	0	unbounded		
choice		1	1		
choice		1	1		
SFT	SFT.CONTENT	1	1		

Data source: <From 'DFDL Test - Parse' view>

Message: HL7 (/workspaces/workspace1/HL7v27DFDLLibrary/GenericHL7.xsd)

Name	Type	Value
HL7		
MSH		
anyHL7Segment		
anyHL7Segment		
anyHL7Segment		
anyHL7Segment		

DFDL Test - Parse: Runs the DFDL parser with the provided physical input data and selected m

Status: Parsing completed: Mon Mar 31 20:53:47 BST 2014

Input

Data: /TestMessages/v27/ADT/A01/A01.txt Browse.

Parsed Input

Characters

```
MSH|^~\&|ADT1|MCM|LABADT|MCM|198808181126|SECU
EVN|A01|198808181123
PID|1||PATID1234^5^M11^ADT1^MR^MCM~123456789^^
```

Parsing completed successfully.

Tips:
- Selecting an element in the DFDL editor will cause the parsed input to focus only on data pertaining to the selected element.
- The view menu on the view toolbar provides options to control how the data is displayed in the view. Click the arrow icon on the toolbar or here to open the menu.
- To view the logical instance that was created by the DFDL parser, click the Open DFDL Logical Instance View toolbar button, or click here.
- To view the trace captured while running the DFDL parser, click the Open DFDL Trace View toolbar button, or click here.

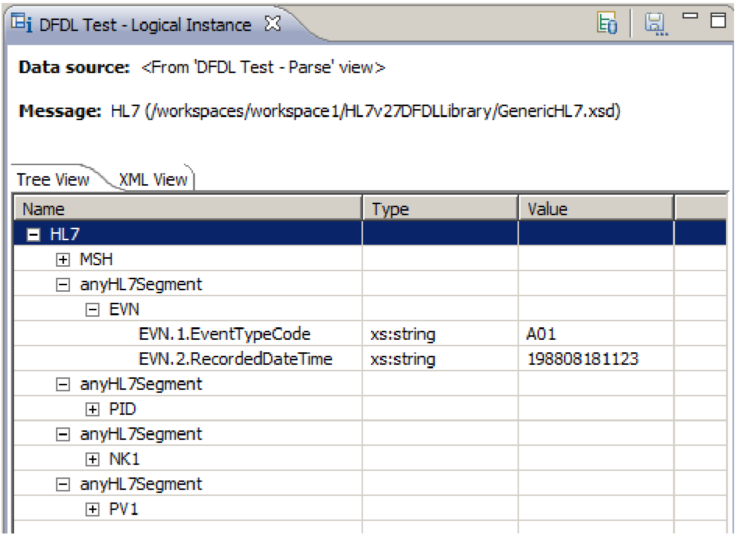
□ Do not display this message again

Selection in DFDL Editor: Selected: HL7 : HL7.CONTENT | Repeating index: 1 | Range in parsed input: 0 - 444

Character Selection In Input: Row: 0 | Column: 0

Byte Selection In Input: Offset: 0 | Length: 0

The view at the base of the window shows how the message's markup has been interpreted. Different colours are used to denote tag values (purple/blue colour) and delimiter values (pink colour). The data itself is shown "normally" as black text on a white background. The hierarchical view in the top right corner shows the logical structure into which the data has been parsed. Note that because we have used the generic HL7 message model, there are entries in the tree structure named anyHL7Segment which wrap the individual segments:

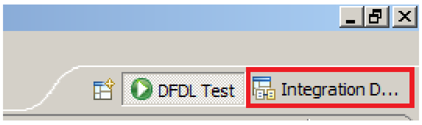


Data source: <From 'DFDL Test - Parse' view>

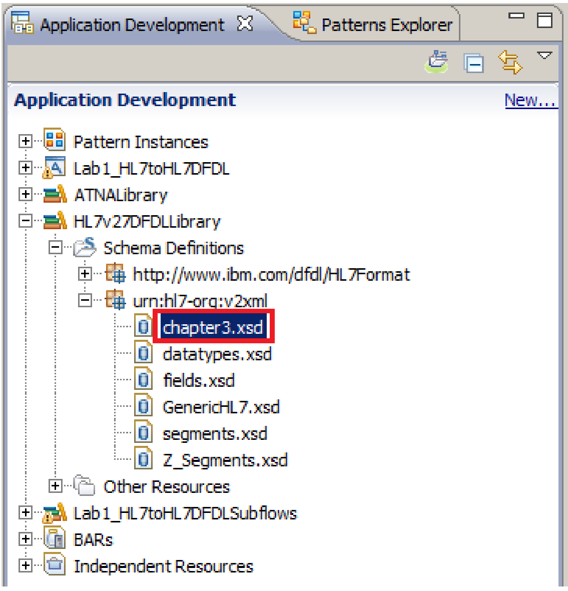
Message: HL7 (/workspaces/workspace1/HL7v27DFDLLibrary/GenericHL7.xsd)

Name	Type	Value
HL7		
MSH		
anyHL7Segment		
EVN		
EVN.1.EventTypeCode	xs:string	A01
EVN.2.RecordedDateTime	xs:string	198808181123
anyHL7Segment		
PID		
anyHL7Segment		
NK1		
anyHL7Segment		
PV1		

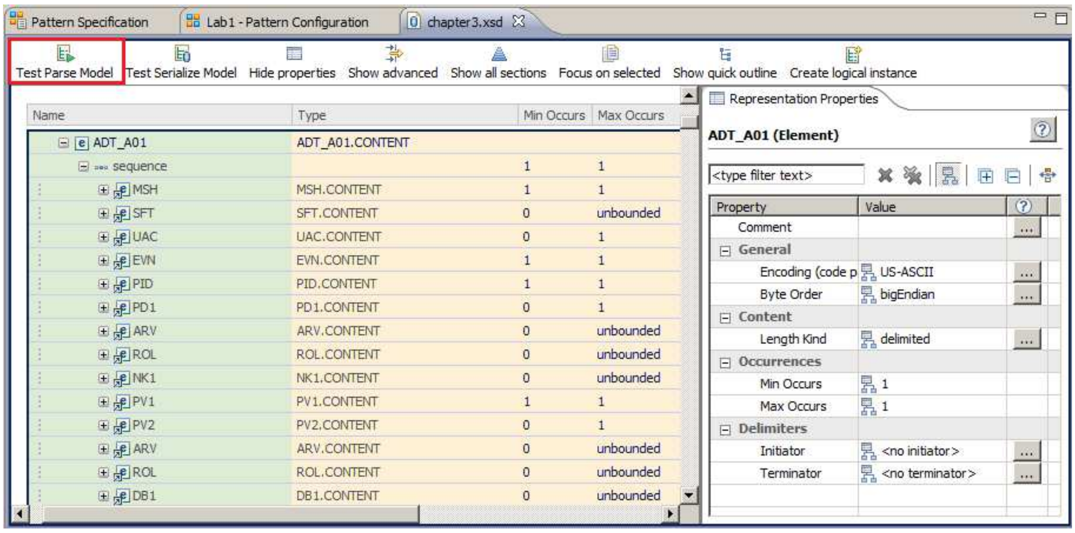
8. Navigate back to the Integration Development perspective using the button in the top right of the Toolkit:



9. Close `GenericHL7.xsd` and double-click `chapter3.xsd` to open this schema instead:



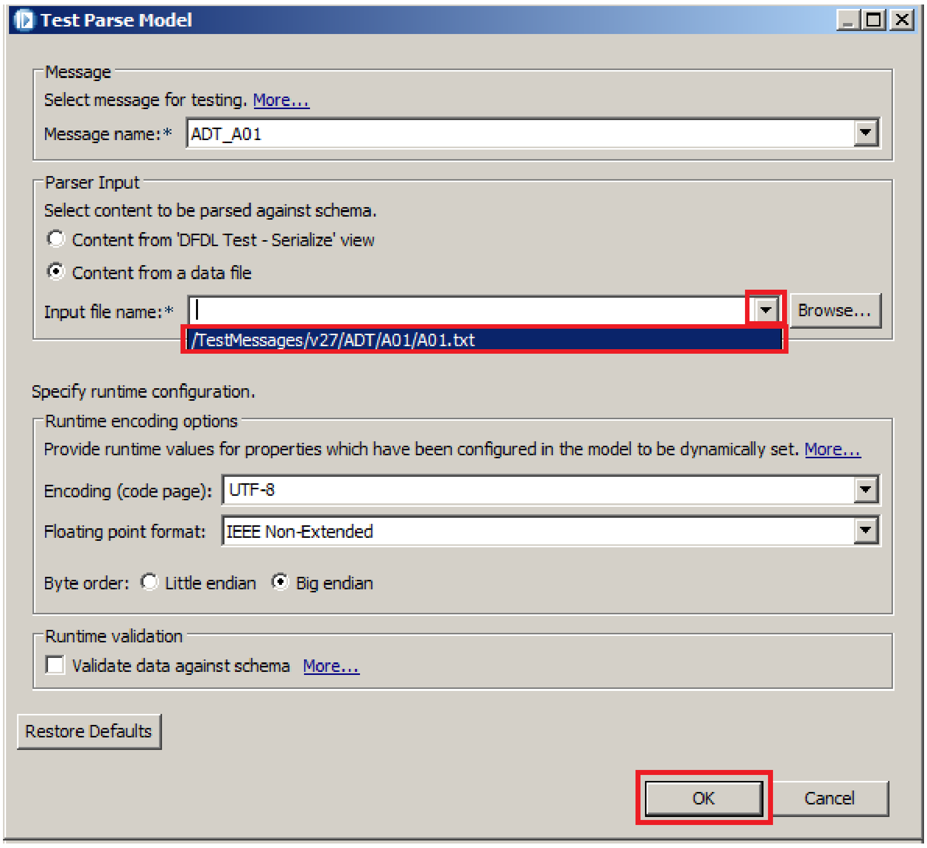
When `chapter3.xsd` opens, scroll down, select the `ADT_A01` message and click the `Test Parse Model` button again:



Name	Type	Min Occurs	Max Occurs
ADT_A01	ADT_A01.CONTENT		
sequence		1	1
MSH	MSH.CONTENT	1	1
SFT	SFT.CONTENT	0	unbounded
UAC	UAC.CONTENT	0	1
EVN	EVN.CONTENT	1	1
PID	PID.CONTENT	1	1
PD1	PD1.CONTENT	0	1
ARV	ARV.CONTENT	0	unbounded
ROL	ROL.CONTENT	0	unbounded
NK1	NK1.CONTENT	0	unbounded
PV1	PV1.CONTENT	1	1
PV2	PV2.CONTENT	0	1
ARV	ARV.CONTENT	0	unbounded
ROL	ROL.CONTENT	0	unbounded
DB1	DB1.CONTENT	0	unbounded

Property	Value
Comment	
General	
Encoding (code p	US-ASCII
Byte Order	bigEndian
Content	
Length Kind	delimited
Occurrences	
Min Occurs	1
Max Occurs	1
Delimiters	
Initiator	<no initiator>
Terminator	<no terminator>

10. Set the `Input file name` field to the same input data as before; this time using the down arrow to recall the path to the test data. Click `OK`:



Test Parse Model

Message

Select message for testing. More...

Message name:* ADT_A01

Parser Input

Select content to be parsed against schema.

○ Content from 'DFDL Test - Serialize' view

● Content from a data file

Input file name:* /TestMessages/v27/ADT/A01/A01.txt

Specify runtime configuration.

Runtime encoding options

Provide runtime values for properties which have been configured in the model to be dynamically set. More...

Encoding (code page): UTF-8

Floating point format: IEEE Non-Extended

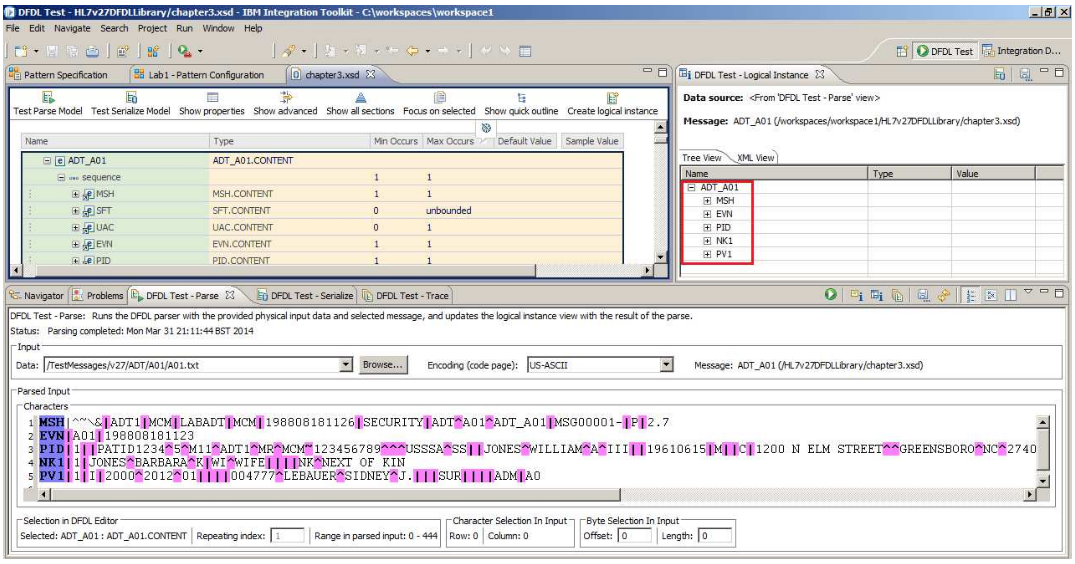
Byte order: ○ Little endian ● Big endian

Runtime validation

□ Validate data against schema More...

As before, dismiss the warning about the impending change of perspective.

11. The resulting parse should again be successful, but this time note that because the message metadata was specifically for the ADT_A01 message type, the parsed hierarchy shown in the top right corner now consists of a straight list of the HL7 segments:



Name	Type	Min Occurs	Max Occurs	Default Value	Sample Value
ADT_A01	ADT_A01.CONTENT				
sequence		1	1		
MSH	MSH.CONTENT	1	1		
SFT	SFT.CONTENT	0	unbounded		
UAC	UAC.CONTENT	0	1		
EVN	EVN.CONTENT	1	1		
PID	PID.CONTENT	1	1		

Data source: <From 'DFDL Test - Parse' view>

Message: ADT_A01 (/workspaces/workspace1/HL7v27DFDLLibrary/chapter3.xsd)

Name	Type	Value
ADT_A01		
MSH		
EVN		
PID		
NK1		
PV1		

DFDL Test - Parse: Runs the DFDL parser with the provided physical input data and selected message, and updates the logical instance view with the result of the parse.

Status: Parsing completed: Mon Mar 31 21:11:44 BST 2014

Input

Data: /TestMessages/v27/ADT/A01/A01.txt Encoding (code page): US-ASCII Message: ADT_A01 (/HL7v27DFDLLibrary/chapter3.xsd)

Parsed Input

Characters

```
MSH|^~\&|ADT1|MCM|LABADT|MCM|198808181126|SECURITY|ADT^A01^ADT_A01|MSG00001-|P|2.7
EVN|A01|198808181123
PID|1||PATID1234^5^M11^ADT1^MR^MCM~123456789^^^USSSA^SS||JONES^WILLIAM^A^III||19610615|M||C|1200 N ELM STREET^^GREENSBORO^NC^2740
NK1|1|JONES^BARBARA^K|WI^WIFE||||NK^NEXT OF KIN
PV1|1|I|2000^2012^01||||004777^LEBAUER^SIDNEY^J.|||SUR||||ADM|A0
```

Selection in DFDL Editor

Selected: ADT_A01 : ADT_A01.CONTENT | Repeating index: 1 | Range in parsed input: 0 - 444

Character Selection In Input: Row: 0 Column: 0

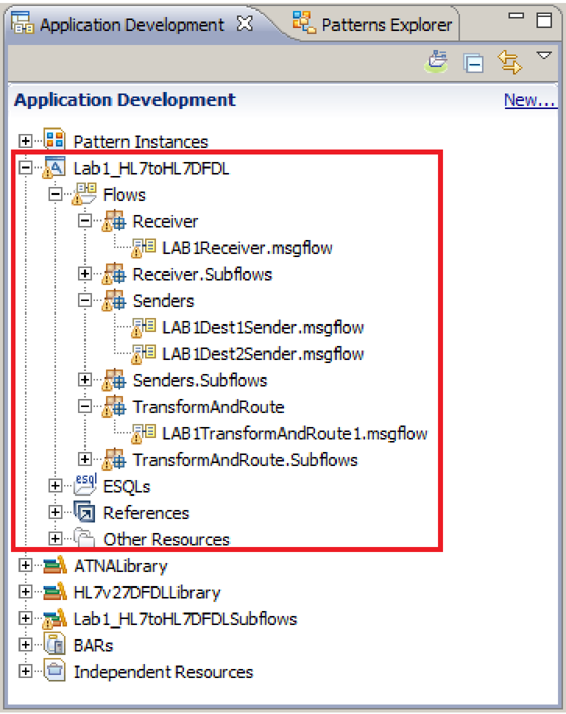
Byte Selection In Input: Offset: 0 Length: 0

Having explored the DFDL HL7 message model, we will now return to explore and test the message flows which were generated by the pattern. Like before, return to the Integration Development perspective using the shortcut at the top right of the Toolkit window. Close all the open files.

4. HL7 to HL7 DFDL Pattern - Generated Application

a. Exploring the Generated Message Flows

1. Now we are back in the Integration Development perspective, in the view on the left, expand the main application (named `Lab1_HL7toHL7DFDL`, if you followed the suggested naming convention):



The application contains several message flows and subflows which are held in appropriately named schema packages. The main four message flows that have been generated are:

Schema: Receiver
`LAB1Receiver`

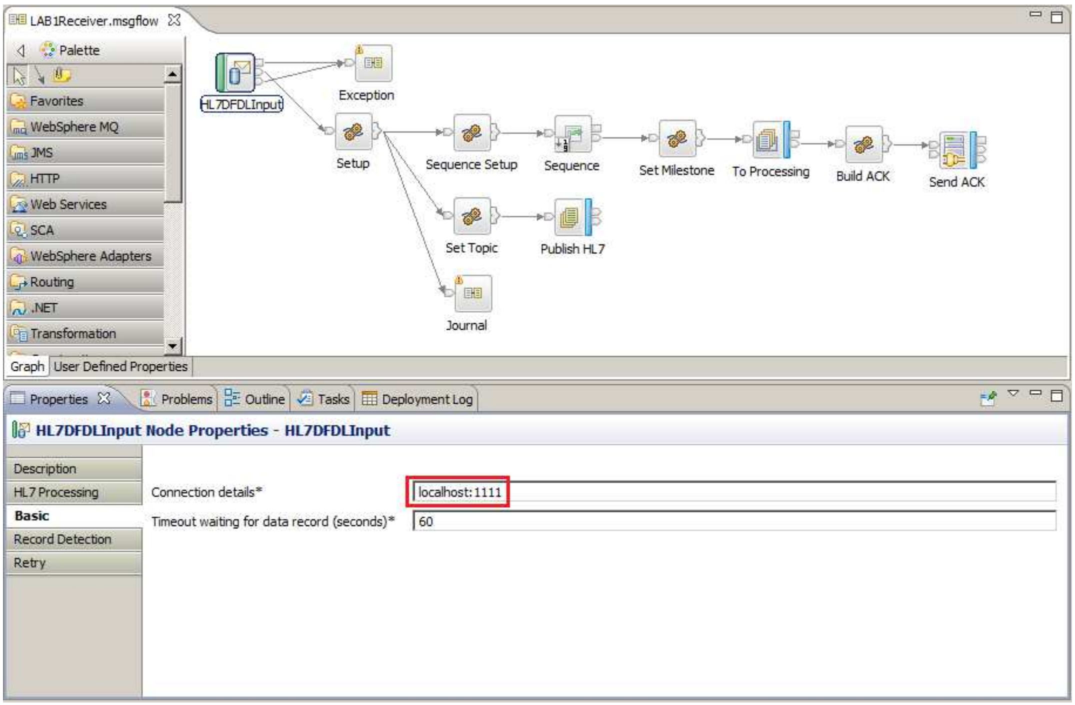
Schema: Senders
`LAB1Dest1Sender`
`LAB1Dest2Sender`

Schema: TransformAndRoute
`LAB1TransformAndRoute1`

2. Open (double-click) the `LAB1Receiver` message flow. This flow receives requests from the HL7 source, using the MLLP protocol. It therefore uses the IIB Healthcare pack's HL7DFDLInput node as its starting point. Click the `HL7DFDLInput` node to see the node's properties. Note that the connection details on the Basic tab are `localhost:1111`. The HL7 Input node is a node provided by the IIB Healthcare Pack. It utilizes the IIB TCP/IP input node, along with some logic for the necessary MLLP protocol checking (eg. HL7 acknowledgements).

   Click each of the node property tabs in turn to see the properties at have been generated. This message flow then sends the request to the Transform and Route message flow. It does this by putting an MQ message to the queue `LAB1.RXF1`. This is done by the node named `To Processing`. Finally, it sends an acknowledgement back to the HL7 requester, using MLLP. This is done by the `Send ACK` node, which is a TCP/IP ServerOutput node.

   Close the message flow.



HL7DFDLInput Node Properties - HL7DFDLInput

Connection details* localhost:1111

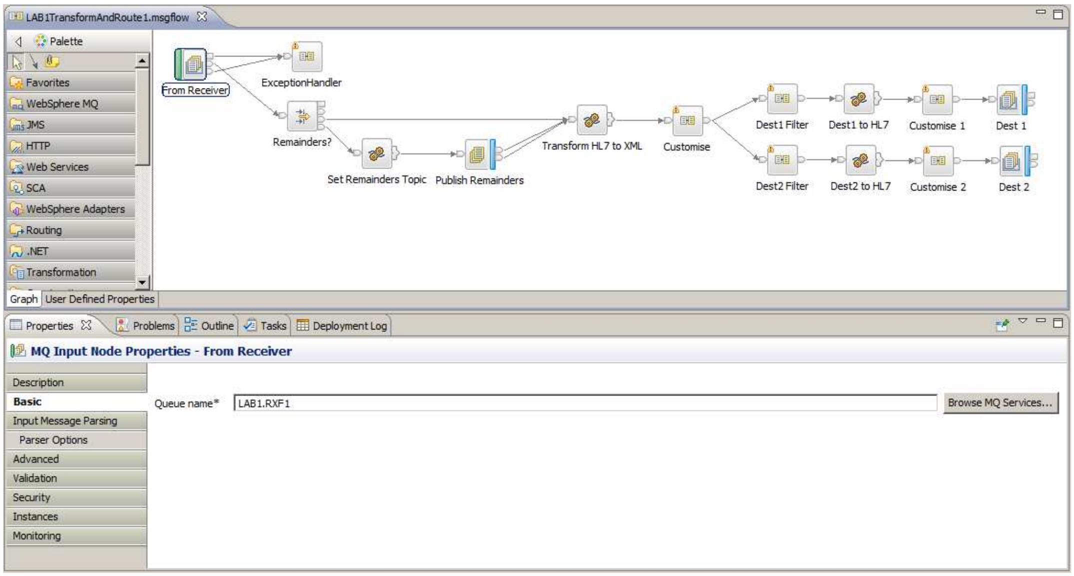
Timeout waiting for data record (seconds)* 60

3. Open the `LAB1TransformAndRoute1` message flow. This flow is started by an MQInput node, which reads the queue `LAB1.RXF1`.

   Since we specified two destinations when generating the pattern, the flow terminates with two routes, terminating with the nodes named `Dest1` and `Dest2`. These are MQOutput nodes, writing to queues `LAB1.DEST1` and `LAB1.DEST2`. Messages on these queues will be processed by the third part of the application, the Sender flows.

   The `LAB1TransformAndRoute1` message flow has several places where you can introduce your own customisation (nodes named `Customise`, `Customise1` and `Customise2`). These nodes are actually subflows which contain a simple passthrough node (you can double-click these nodes to see the subflow, and close it when you're done). These customizable subflows can be used to provide specific transformations, or any other logic that you need for your own scenarios.

   Close the message flow.

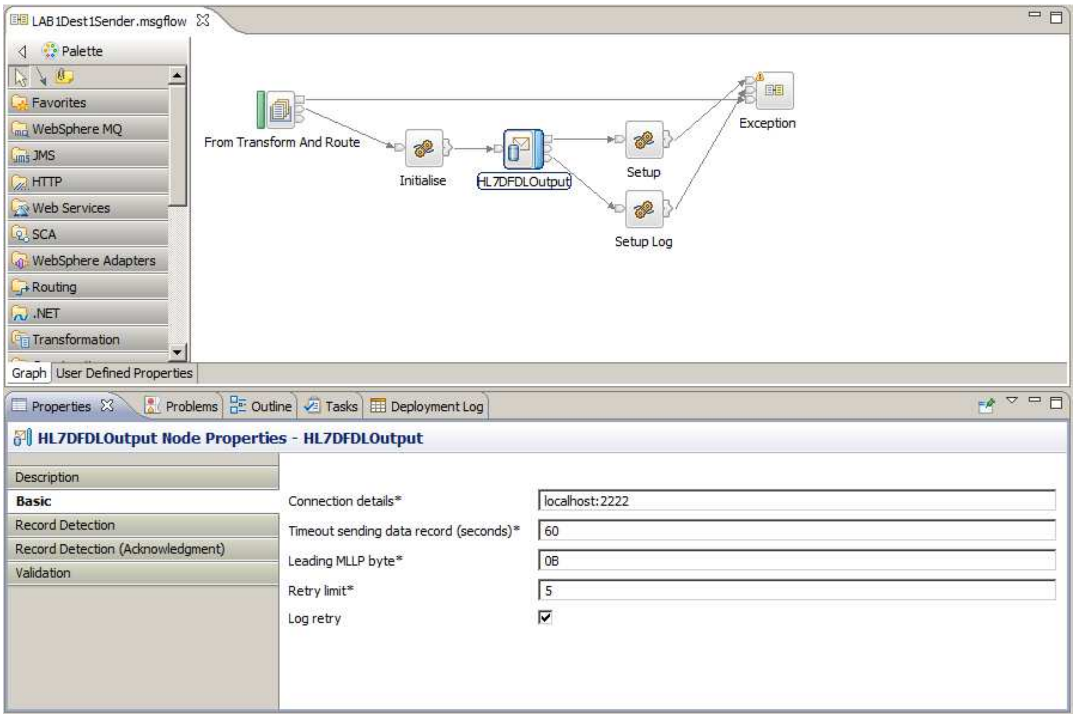


MQ Input Node Properties - From Receiver

Queue name* LAB1.RXF1

4. Open the message flow named `Lab1Dest1Sender`. This flow sends the HL7 request to the first of the destinations that we requested in the pattern. It takes its input from the MQ queue `LAB1.DEST1`, and uses an HL7 Output node to communicate with the destination application. The node checks for an acknowledgement from that destination. If no acknowledgement is received, a Timeout mechanism is activated.

   Communication with the second destination that we specified in the pattern, is done via the message flow `Lab1Dest2Sender` which this has the same logic as Dest1. Explore the flow for yourself and close the message flows when you’re finished.



HL7DFDLOutput Node Properties - HL7DFDLOutput

Connection details* localhost:2222

Timeout sending data record (seconds)* 60

Leading MLLP byte* 0B

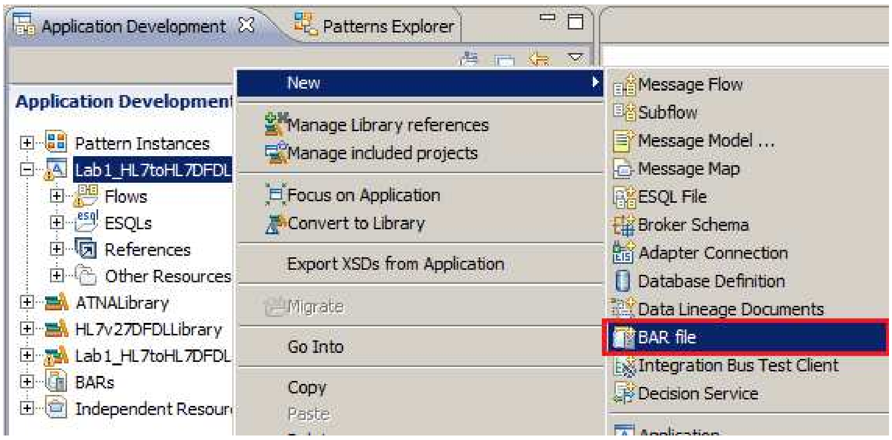
Retry limit* 5

Log retry ☑

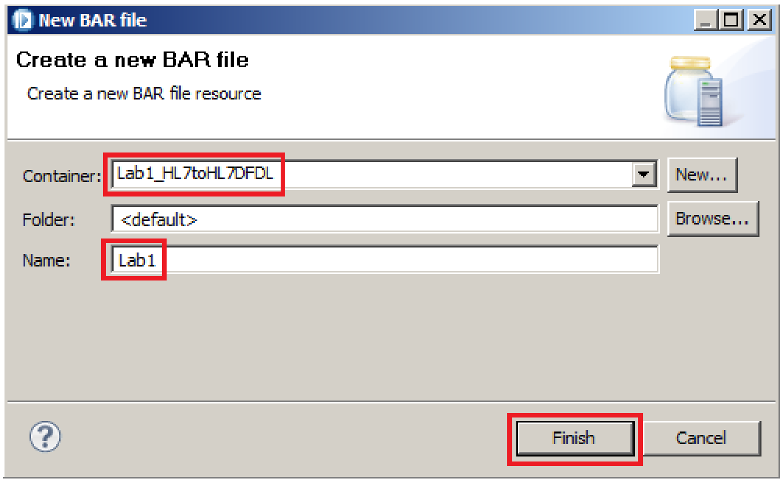
5. Deploying the Applications

a. Creating and Deploying a Broker Archive File

1. Right click the generated application `Lab1_HL7toHL7DFDL` and select `New`, `BAR file`



2. Specify the Container of `Lab1_HL7toHL7DFDL` and name the Broker Archive (BAR for short) file `Lab1` or something of your own choosing (the name is not important) and click `Finish`.



Create a new BAR file

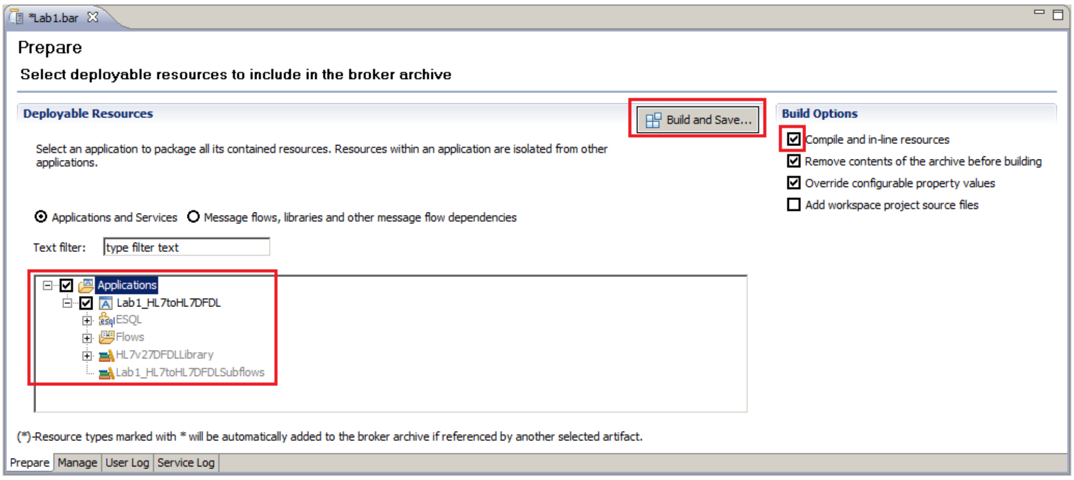
Create a new BAR file resource

Container: Lab1_HL7toHL7DFDL

Folder: <default>

Name: Lab1

3. The newly created BAR file will be opened in the BAR editor. Select the checkbox option to `Compile and in-line resources`, select the `Lab1_HL7toHL7DFDL` application and then click the `Build and Save` button.



*Lab1.bar

Prepare

Select deployable resources to include in the broker archive

Deployable Resources

Select an application to package all its contained resources. Resources within an application are isolated from other applications.

● Applications and Services ○ Message flows, libraries and other message flow dependencies

Text filter: type filter text

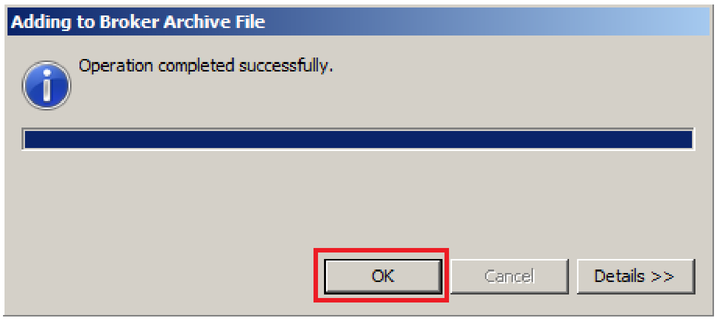
- ☑ Applications
  - ☑ Lab1_HL7toHL7DFDL
    - ESQL
    - Flows
    - HL7v27DFDLLibrary
    - Lab1_HL7toHL7DFDLSubflows

Build Options

☑ Compile and in-line resources
☑ Remove contents of the archive before building
☑ Override configurable property values
☐ Add workspace project source files

(*)-Resource types marked with * will be automatically added to the broker archive if referenced by another selected artifact.

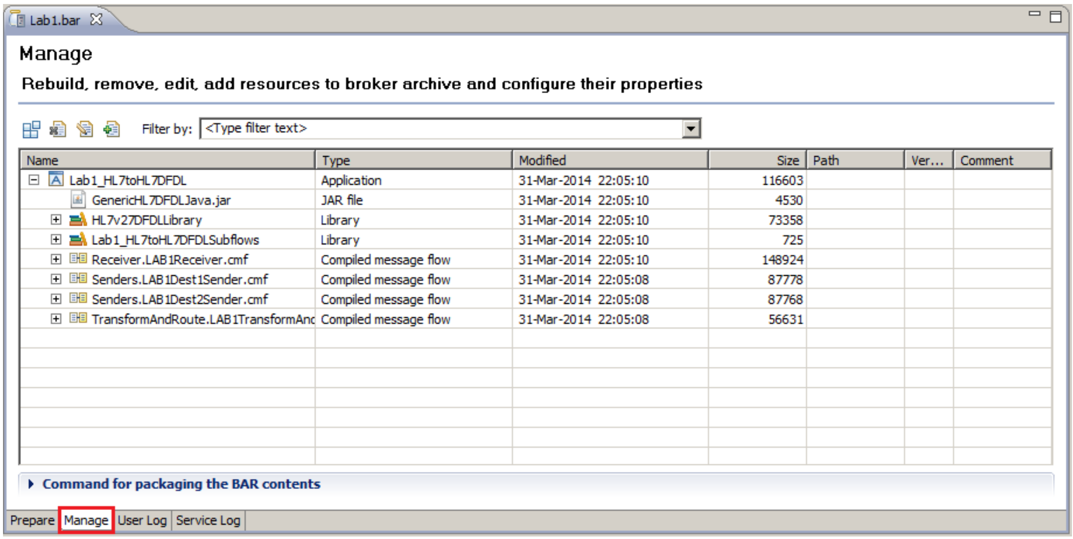
When the operation completes, click `OK`:



Adding to Broker Archive File

Operation completed successfully.

4. Click the Manage tab and you will see that the application's dependent library and subflows have automatically been built into the BAR file as well:



Lab1.bar

Manage

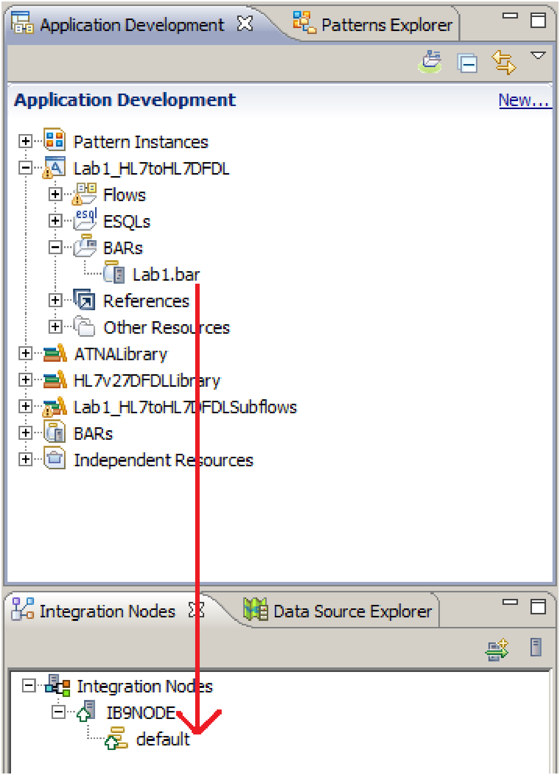
Rebuild, remove, edit, add resources to broker archive and configure their properties

Filter by: <Type filter text>

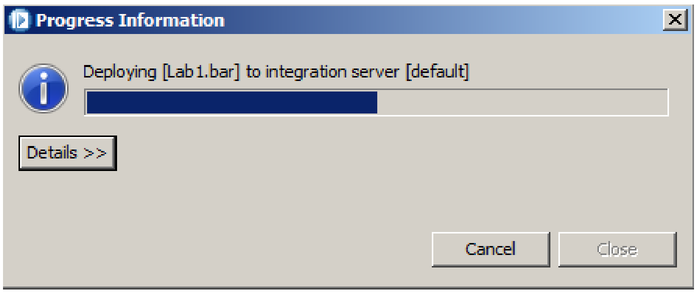
Name	Type	Modified	Size	Path	Ver...	Comment
Lab1_HL7toHL7DFDL	Application	31-Mar-2014 22:05:10	116603			
GenericHL7DFDLJava.jar	JAR file	31-Mar-2014 22:05:10	4530			
HL7v27DFDLLibrary	Library	31-Mar-2014 22:05:10	73358			
Lab1_HL7toHL7DFDLSubflows	Library	31-Mar-2014 22:05:10	725			
Receiver.LAB1Receiver.cmf	Compiled message flow	31-Mar-2014 22:05:10	148924			
Senders.LAB1Dest1Sender.cmf	Compiled message flow	31-Mar-2014 22:05:08	87778			
Senders.LAB1Dest2Sender.cmf	Compiled message flow	31-Mar-2014 22:05:08	87768			
TransformAndRoute.LAB1TransformAnc	Compiled message flow	31-Mar-2014 22:05:08	56631			

Command for packaging the BAR contents

5. Now deploy the BAR file by dragging and dropping it on top of the default server (alternatively you can right-click the file and select `Deploy`).

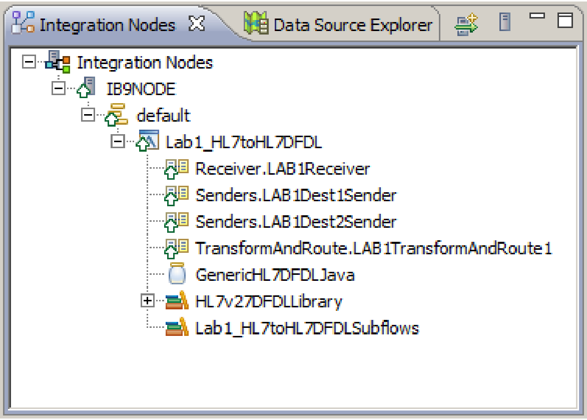


Deployment will take about a minute complete, so please be patient!



Deploying [Lab1.bar] to integration server [default]

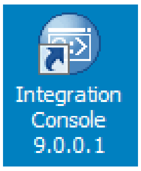
6. When the deployment is complete, you will see the deployed artifacts in the Integration Nodes view in the bottom left corner of the Toolkit:



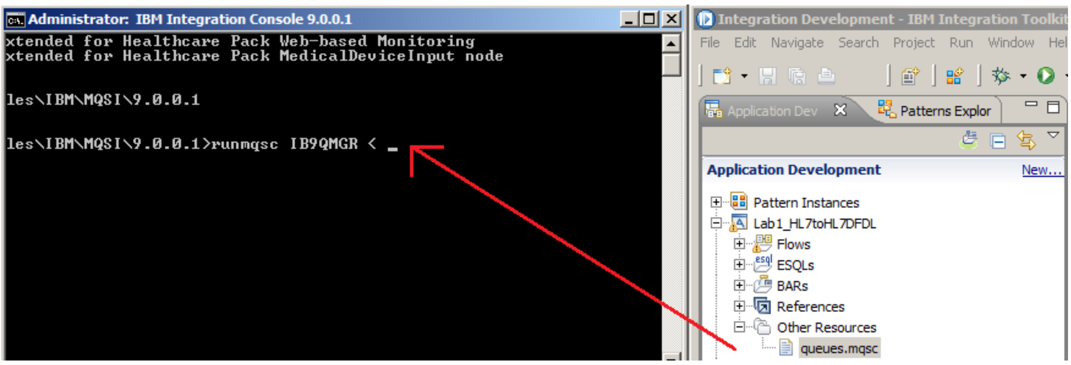
- Integration Nodes
  - IB9NODE
    - default
      - Lab1_HL7toHL7DFDL
        - Receiver.LAB1Receiver
        - Senders.LAB1Dest1Sender
        - Senders.LAB1Dest2Sender
        - TransformAndRoute.LAB1TransformAndRoute1
        - GenericHL7DFDLJava
        - HL7v27DFDLLibrary
        - Lab1_HL7toHL7DFDLSubflows

To prepare for testing, close the open views of any message flows or BAR files that are open in the Toolkit. This will make it easier to view further artifacts that will be opened.

7. Finally, you need to define several MQ queues. The queues required for the specific HL7 message flows need to be created. We configured the pattern instance to generate the necessary MQSC script, and this script contains the necessary statements. If you are using the VMWare image, from the desktop launch the Integration Console using this shortcut:



At the prompt type the instruction **`runmqsc IB9QMGR <`** …and then to complete the instruction, drag and drop the MQSC script from the Integration Toolkit:

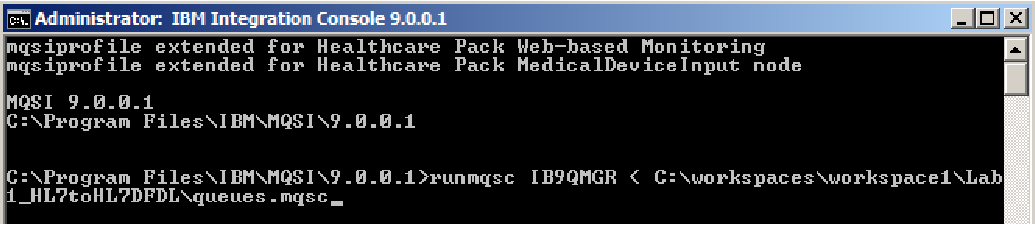


```
xtended for Healthcare Pack Web-based Monitoring
xtended for Healthcare Pack MedicalDeviceInput node

les\IBM\MQSI\9.0.0.1

les\IBM\MQSI\9.0.0.1>runmqsc IB9QMGR < _
```

This drag and drop approach will avoid you having to type out the full file system path to the mqsc script. Hit return on the command and the script will execute:



```
mqsiprofile extended for Healthcare Pack Web-based Monitoring
mqsiprofile extended for Healthcare Pack MedicalDeviceInput node

MQSI 9.0.0.1
C:\Program Files\IBM\MQSI\9.0.0.1


C:\Program Files\IBM\MQSI\9.0.0.1>runmqsc IB9QMGR < C:\workspaces\workspace1\Lab
1_HL7toHL7DFDL\queues.mqsc_
```

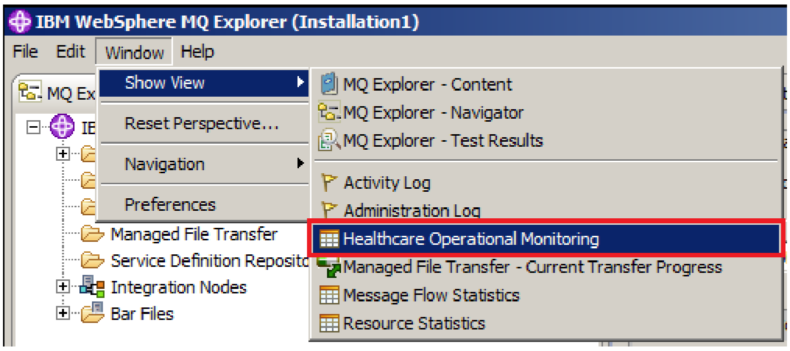
Note: If you decide to type the file system path, you may have difficulties with the slash character if you are running the VMWare image on a host machine which does not have the keyboard set to be UK English. (If you are using a machine with a US keyboard setting on the host, then you can work around this for the slash character, by holding the {alt} key while entering “92” on the numeric keypad).

6. Testing the HL7 Application

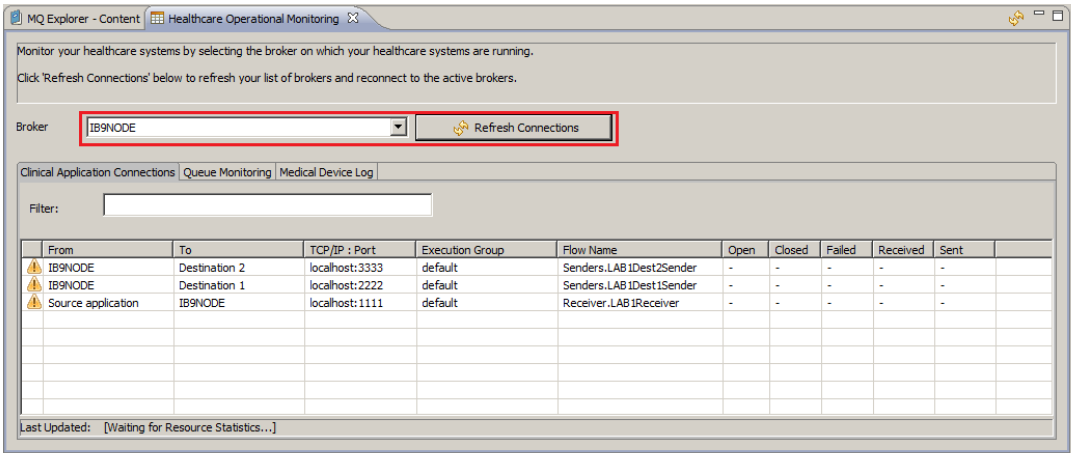
a. The Integration Explorer Healthcare Operational Monitoring View

1. We are about to begin testing the HL7 to HL7 DFDL Pattern which has been deployed, but before doing so, we will explore the IIB Healthcare Pack's real-time monitoring capabilities. If you are using the VMWare image, from the desktop launch the Integration Explorer using this shortcut:

2. When the Integration Explorer is launched, open the Healthcare Operational Monitoring view from the `Window`, `Show View` menu.



3. In the `Healthcare Operational Monitoring` view, make sure that the `IB9NODE` is selected, then click the Refresh Connections button. The message flows which make up the deployed pattern should be displayed in the `Clinical Application Connections` view. We have not yet sent any data through the flows, or connected end-point applications, so you will see warning symbols at the left of each row, and no values in the `Open`, `Closed`, `Failed`, `Received`, and `Sent` columns:



Monitor your healthcare systems by selecting the broker on which your healthcare systems are running.

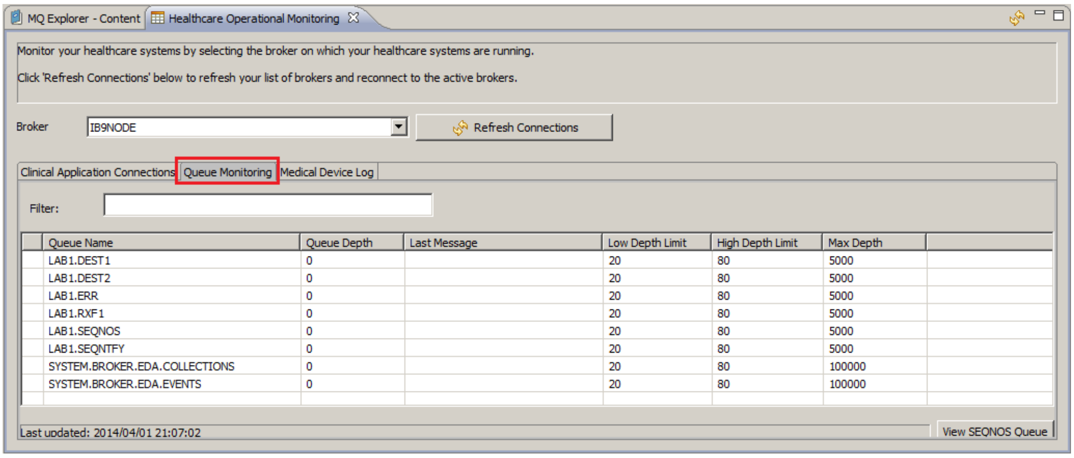
Click 'Refresh Connections' below to refresh your list of brokers and reconnect to the active brokers.

Broker: IB9NODE

	From	To	TCP/IP : Port	Execution Group	Flow Name	Open	Closed	Failed	Received	Sent
⚠	IB9NODE	Destination 2	localhost:3333	default	Senders.LAB1Dest2Sender	-	-	-	-	-
⚠	IB9NODE	Destination 1	localhost:2222	default	Senders.LAB1Dest1Sender	-	-	-	-	-
⚠	Source application	IB9NODE	localhost:1111	default	Receiver.LAB1Receiver	-	-	-	-	-

Last Updated: [Waiting for Resource Statistics...]

4. Now click on the Queue Monitoring tab and a filtered view of the queues associated with the pattern will be displayed:



Monitor your healthcare systems by selecting the broker on which your healthcare systems are running.

Click 'Refresh Connections' below to refresh your list of brokers and reconnect to the active brokers.

Broker: IB9NODE

Filter:

Queue Name	Queue Depth	Last Message	Low Depth Limit	High Depth Limit	Max Depth
LAB1.DEST1	0		20	80	5000
LAB1.DEST2	0		20	80	5000
LAB1.ERR	0		20	80	5000
LAB1.RXF1	0		20	80	5000
LAB1.SEQNOS	0		20	80	5000
LAB1.SEQNTFY	0		20	80	5000
SYSTEM.BROKER.EDA.COLLECTIONS	0		20	80	100000
SYSTEM.BROKER.EDA.EVENTS	0		20	80	100000

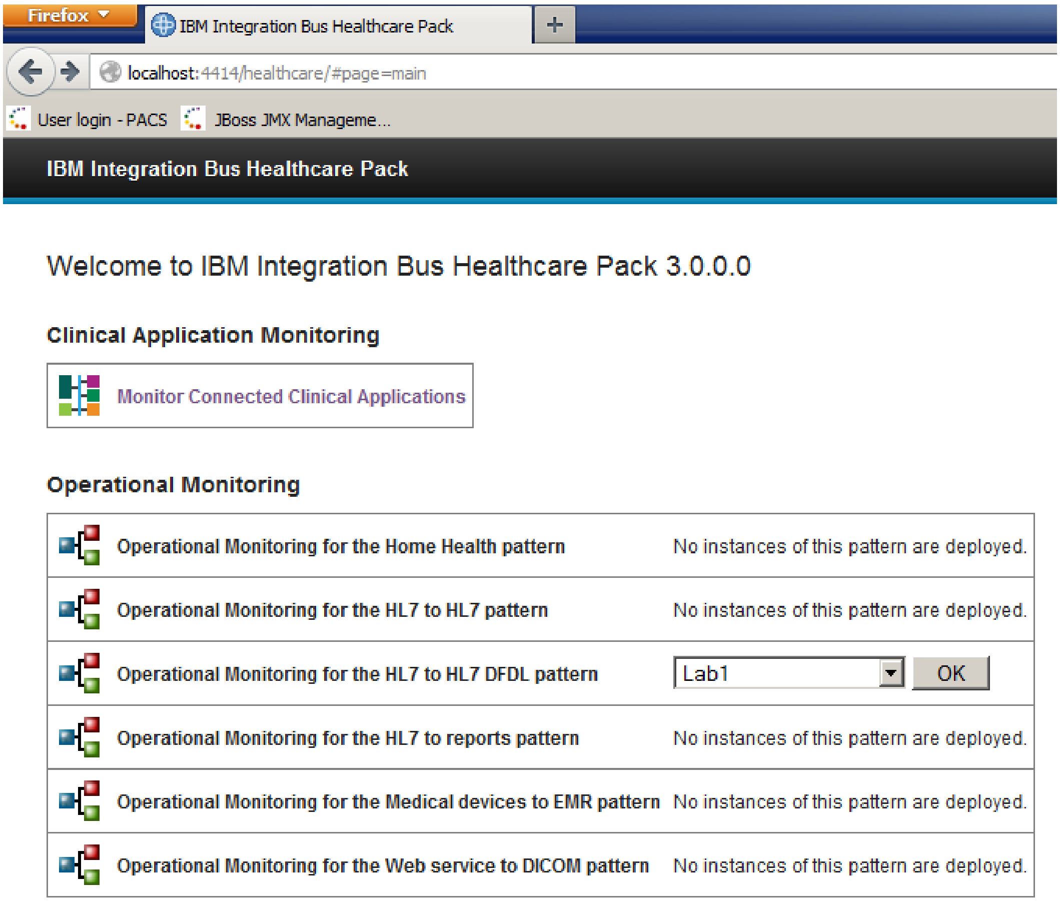
Last updated: 2014/04/01 21:07:02

At this stage all the queue depths should be zero because we have not yet sent test data through the pattern. You can return to the Clinical Application Connections and Queue Monitoring views in the Integration Explorer whilst running the tests in a moment, but before doing so we will also look at the available monitoring views in the Web UI.

b. Exploring the Web UI Clinical Application Monitoring View

1. Launch a web browser. If you are using the VMWare image, there is a link to the Mozilla Firefox browser on the desktop:

The Firefox browser's homepage has been configured to point at the IIB Healthcare pack's homepage. Your node should be started, so when the browser launches you should see the following:


Firefox
IBM Integration Bus Healthcare Pack
+
localhost:4414/healthcare/#page=main
User login - PACS
JBoss JMX Manageme...

IBM Integration Bus Healthcare Pack

Welcome to IBM Integration Bus Healthcare Pack 3.0.0.0

Clinical Application Monitoring

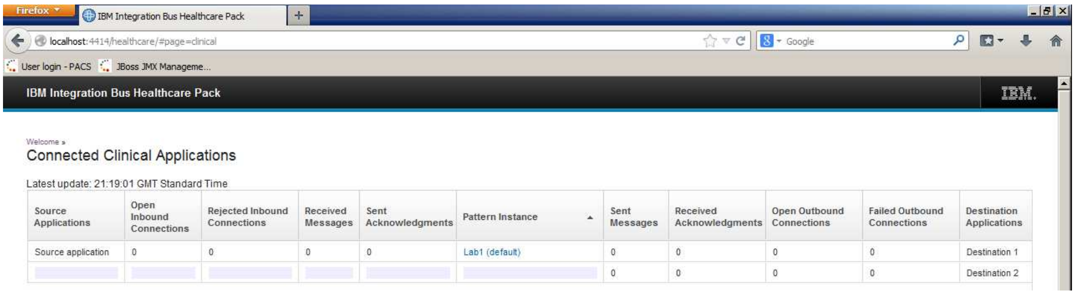
Monitor Connected Clinical Applications

Operational Monitoring

Operational Monitoring for the Home Health pattern	No instances of this pattern are deployed.
Operational Monitoring for the HL7 to HL7 pattern	No instances of this pattern are deployed.
Operational Monitoring for the HL7 to HL7 DFDL pattern	Lab1 ▾ OK
Operational Monitoring for the HL7 to reports pattern	No instances of this pattern are deployed.
Operational Monitoring for the Medical devices to EMR pattern	No instances of this pattern are deployed.
Operational Monitoring for the Web service to DICOM pattern	No instances of this pattern are deployed.

Click the hyperlink named `Monitor Connected Clinical Applications`.

2. The `Connected Clinical Applications` view should display the two destinations, `Destination1` and `Destination2` grouped against the `Source application`.



Firefox
IBM Integration Bus Healthcare Pack
localhost:4414/healthcare/#page=clinical
Google
User login - PACS
JBoss JMX Manageme...
IBM Integration Bus Healthcare Pack
IBM.
Welcome »

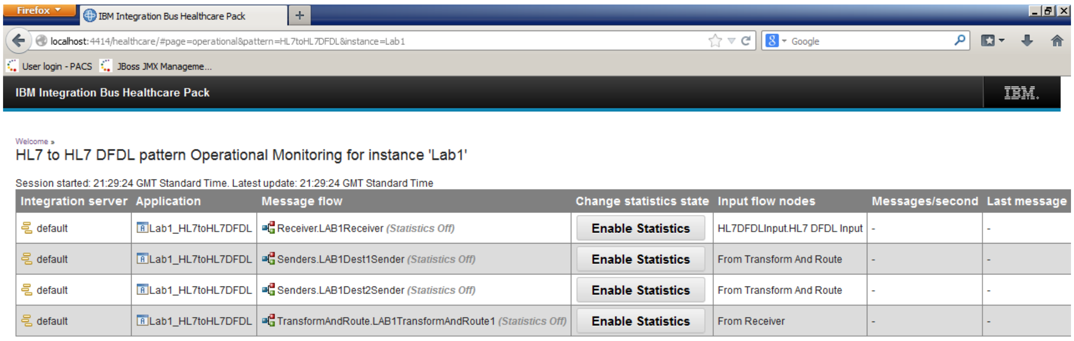
Connected Clinical Applications

Latest update: 21:19:01 GMT Standard Time

Source Applications	Open Inbound Connections	Rejected Inbound Connections	Received Messages	Sent Acknowledgments	Pattern Instance	Sent Messages	Received Acknowledgments	Open Outbound Connections	Failed Outbound Connections	Destination Applications
Source application	0	0	0	0	Lab1 (default)	0	0	0	0	Destination 1
						0	0	0	0	Destination 2

We are yet to send data through the flows, so for the moment the counters should show the value 0.

3. The Pattern Instance column contains a hyperlink with the pattern instance name, and the server (execution group) to which it is deployed in brackets. In our case this is named `Lab1 (default)`. If you click this link you will be taken to the `Operational Monitoring View` for the `Lab1` pattern instance of the HL7 to HL7 DFDL Pattern type. This view shows information about the message flows which make up the pattern:



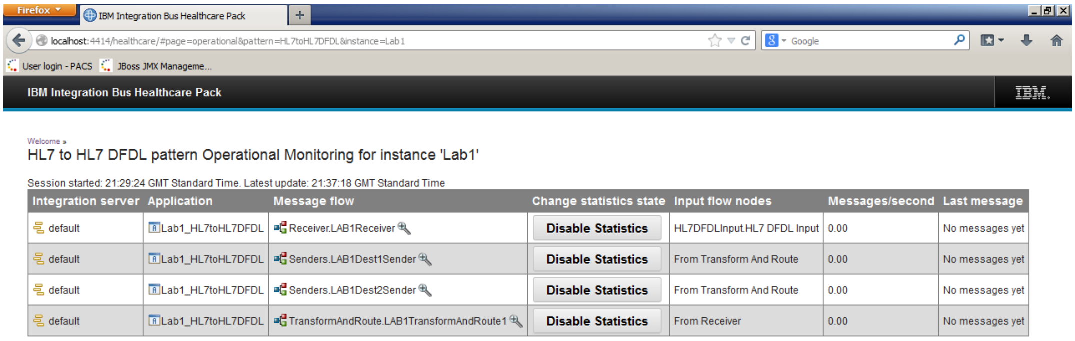
Firefox
IBM Integration Bus Healthcare Pack
localhost:4414/healthcare/#page=operational&pattern=HL7toHL7DFDL&instance=Lab1
Google
User login - PACS
JBoss JMX Manageme...
IBM Integration Bus Healthcare Pack
IBM.
Welcome »

HL7 to HL7 DFDL pattern Operational Monitoring for instance 'Lab1'

Session started: 21:29:24 GMT Standard Time. Latest update: 21:29:24 GMT Standard Time

Integration server	Application	Message flow	Change statistics state	Input flow nodes	Messages/second	Last message
default	Lab1_HL7toHL7DFDL	Receiver.LAB1Receiver *(Statistics Off)*	Enable Statistics	HL7DFDLInput.HL7 DFDL Input	-	-
default	Lab1_HL7toHL7DFDL	Senders.LAB1Dest1Sender *(Statistics Off)*	Enable Statistics	From Transform And Route	-	-
default	Lab1_HL7toHL7DFDL	Senders.LAB1Dest2Sender *(Statistics Off)*	Enable Statistics	From Transform And Route	-	-
default	Lab1_HL7toHL7DFDL	TransformAndRoute.LAB1TransformAndRoute1 *(Statistics Off)*	Enable Statistics	From Receiver	-	-

4. Click each of the buttons labelled `Enable Statistics`. This action ensures that the message flows will publish IIB Accounting & Statistics information when we exercise them in a moment.

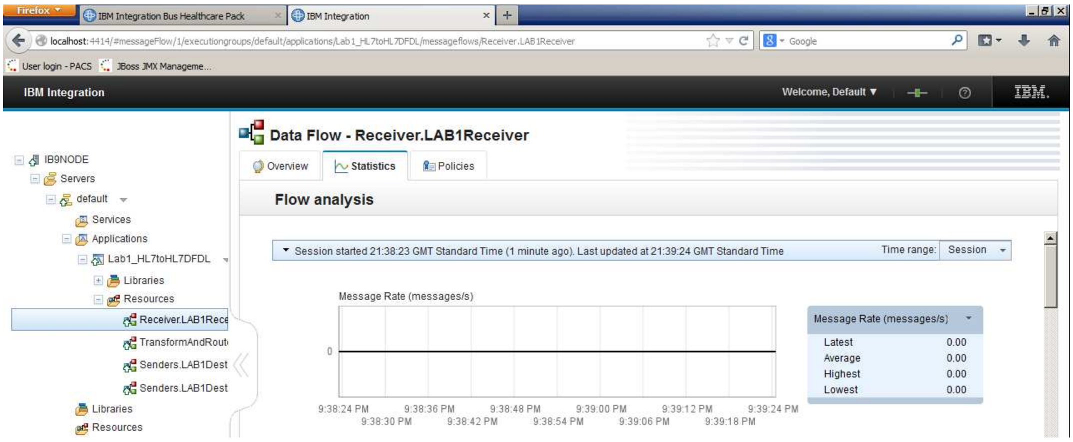

Firefox
IBM Integration Bus Healthcare Pack
localhost:4414/healthcare/#page=operational&pattern=HL7toHL7DFDL&instance=Lab1
Google
User login - PACS
JBoss JMX Manageme...
IBM Integration Bus Healthcare Pack
IBM.
Welcome »

HL7 to HL7 DFDL pattern Operational Monitoring for instance 'Lab1'

Session started: 21:29:24 GMT Standard Time. Latest update: 21:37:18 GMT Standard Time

Integration server	Application	Message flow	Change statistics state	Input flow nodes	Messages/second	Last message
default	Lab1_HL7toHL7DFDL	Receiver.LAB1Receiver	Disable Statistics	HL7DFDLInput.HL7 DFDL Input	0.00	No messages yet
default	Lab1_HL7toHL7DFDL	Senders.LAB1Dest1Sender	Disable Statistics	From Transform And Route	0.00	No messages yet
default	Lab1_HL7toHL7DFDL	Senders.LAB1Dest2Sender	Disable Statistics	From Transform And Route	0.00	No messages yet
default	Lab1_HL7toHL7DFDL	TransformAndRoute.LAB1TransformAndRoute1	Disable Statistics	From Receiver	0.00	No messages yet

Once you have clicked the buttons and the statistics state has changed, you will see that the Message flow column entries will be updated with an icon signifying a picture of a magnifying glass. If you click one of these links a new browser tab will be opened containing a standard IIB Web UI view which contains a more technical detailed view of flow activity:



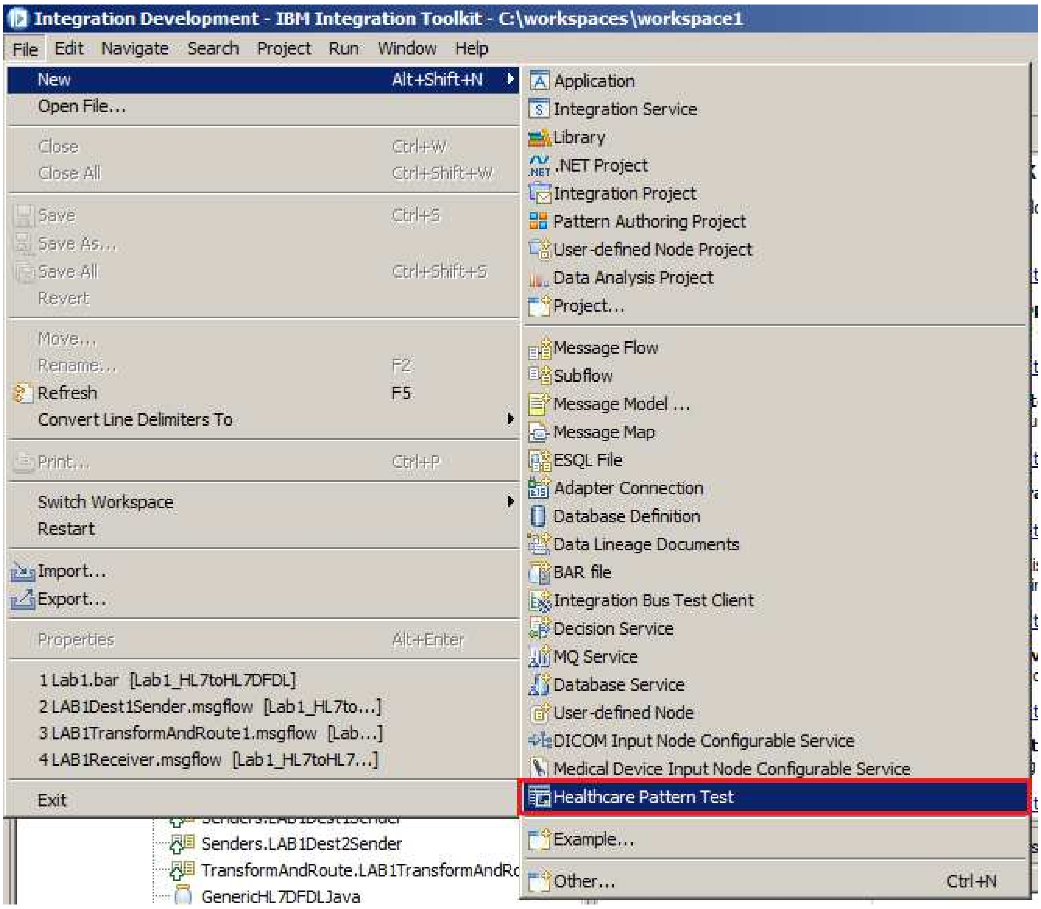
Session started 21:38:23 GMT Standard Time (1 minute ago). Last updated at 21:39:24 GMT Standard Time

Message Rate (messages/s)	
Latest	0.00
Average	0.00
Highest	0.00
Lowest	0.00

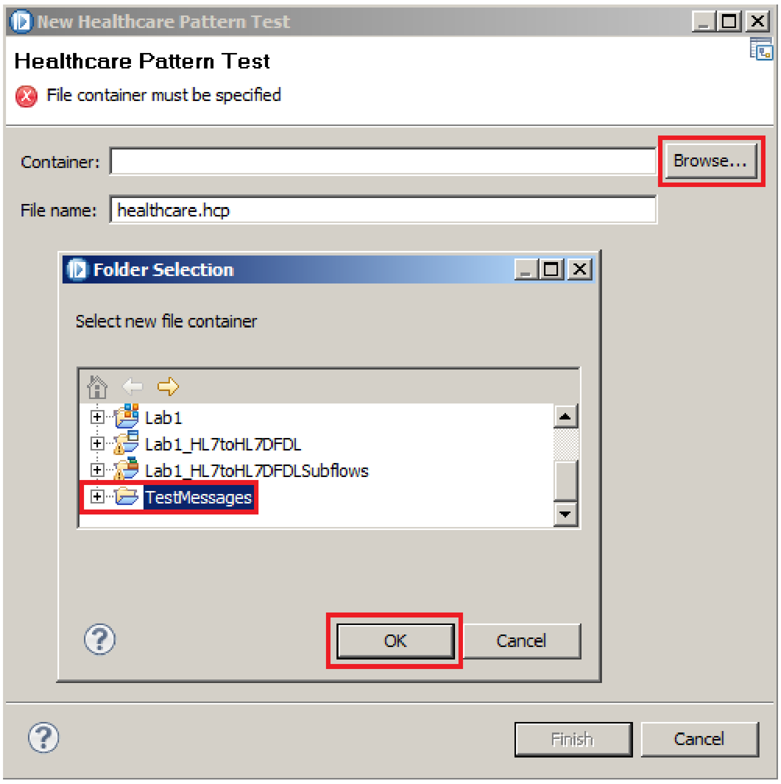
c. Testing the Pattern using the Integration Toolkit built-in test tools

1. The IIB Healthcare Pack includes a built-in test tool designed specifically for scenarios involving HL7 messages sent and received over the MLLP socket protocol.

   In the IIB Toolkit, from the File menu, choose `New`, `Healthcare Pattern Test`.



2. In the resulting dialog, click the `Browse` button to provide a Container in which the Healthcare Pattern Test file will be saved. Select the `TestMessages` project (You can define a separate project to hold this resource, but it is advisable not to choose the Lab1_HL7toHL7DFDL project, because this will inhibit regeneration of that project by the pattern, which you may wish to do later).



New Healthcare Pattern Test

Healthcare Pattern Test

File container must be specified

Container:

File name: healthcare.hcp

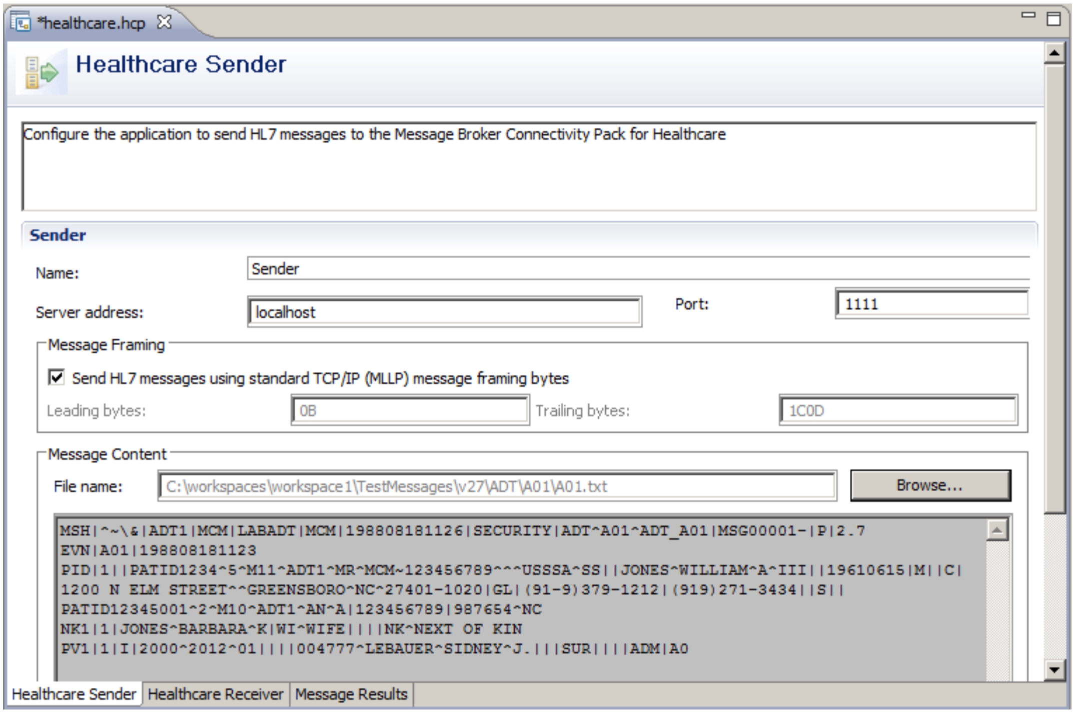
Folder Selection

Select new file container

Click `OK` and `Finish` to complete.

3. The `healthcare.hcp` editor will open. The healthcare Test tool provides three primary screens (`Healthcare Sender`, `Healthcare Receiver` and `Message Results`) as shown below. The `Healthcare Sender` tab provides the place to enter details of the HL7 message and the sender connection details.

   Use the `Browse` button to load a sample HL7 message. The example below shows an ADT_A01 message. These sample messages are stored in your workspace directory under the folder `\TestMessages\v27\ADT`.



*healthcare.hcp

Healthcare Sender

Configure the application to send HL7 messages to the Message Broker Connectivity Pack for Healthcare

Sender

Name: Sender

Server address: localhost

Port: 1111

Message Framing

☑ Send HL7 messages using standard TCP/IP (MLLP) message framing bytes

Leading bytes: 0B

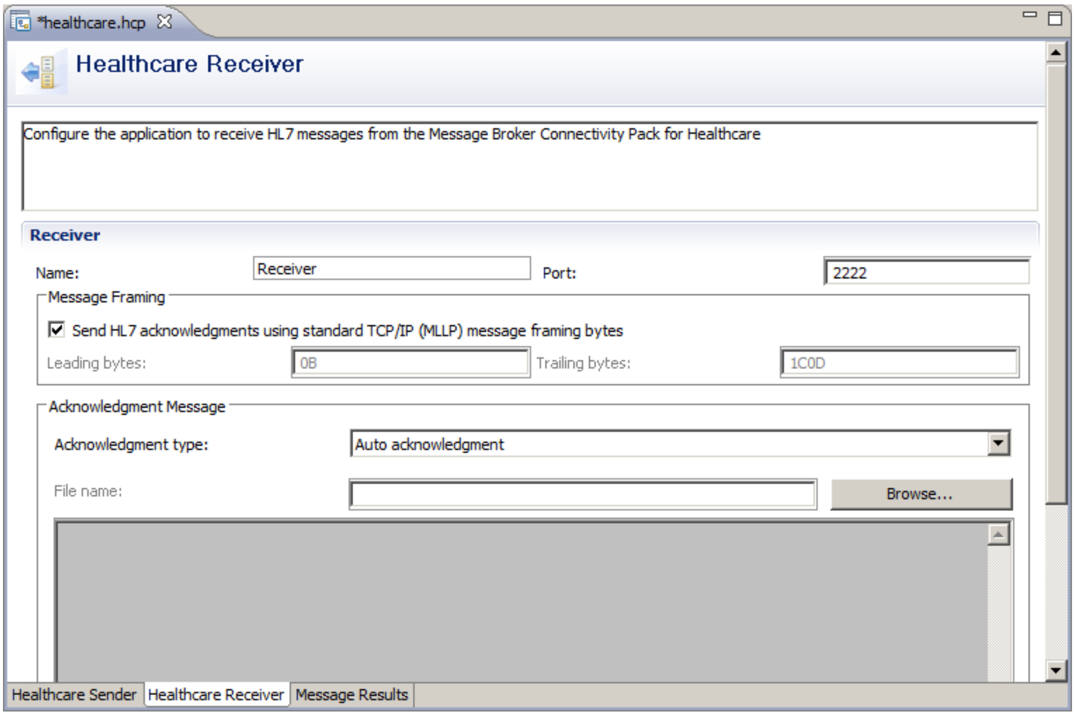
Trailing bytes: 1C0D

Message Content

File name: C:\workspaces\workspace1\TestMessages\v27\ADT\A01\A01.txt

```
MSH|^~\&|ADT1|MCM|LABADT|MCM|198808181126|SECURITY|ADT^A01^ADT_A01|MSG00001-|P|2.7
EVN|A01|198808181123
PID|1||PATID1234^5^M11^ADT1^MR^MCM~123456789^^^USSSA^SS||JONES^WILLIAM^A^III||19610615|M||C|
1200 N ELM STREET^^GREENSBORO^NC^27401-1020|GL|(91-9)379-1212|(919)271-3434||S||
PATID12345001^2^M10^ADT1^AN^A|123456789|987654^NC
NK1|1|JONES^BARBARA^K|WI^WIFE||||NK^NEXT OF KIN
PV1|1|I|2000^2012^01||||004777^LEBAUER^SIDNEY^J.|||SUR||||ADM|A0
```

4. Switch to the `Healthcare Receiver` tab. This will have been pre-populated to use the port `2222`. This will be fine for this lab, so do not make any changes here.



Healthcare Receiver

Configure the application to receive HL7 messages from the Message Broker Connectivity Pack for Healthcare

Receiver

Name: Receiver Port: 2222

Message Framing

☑ Send HL7 acknowledgments using standard TCP/IP (MLLP) message framing bytes

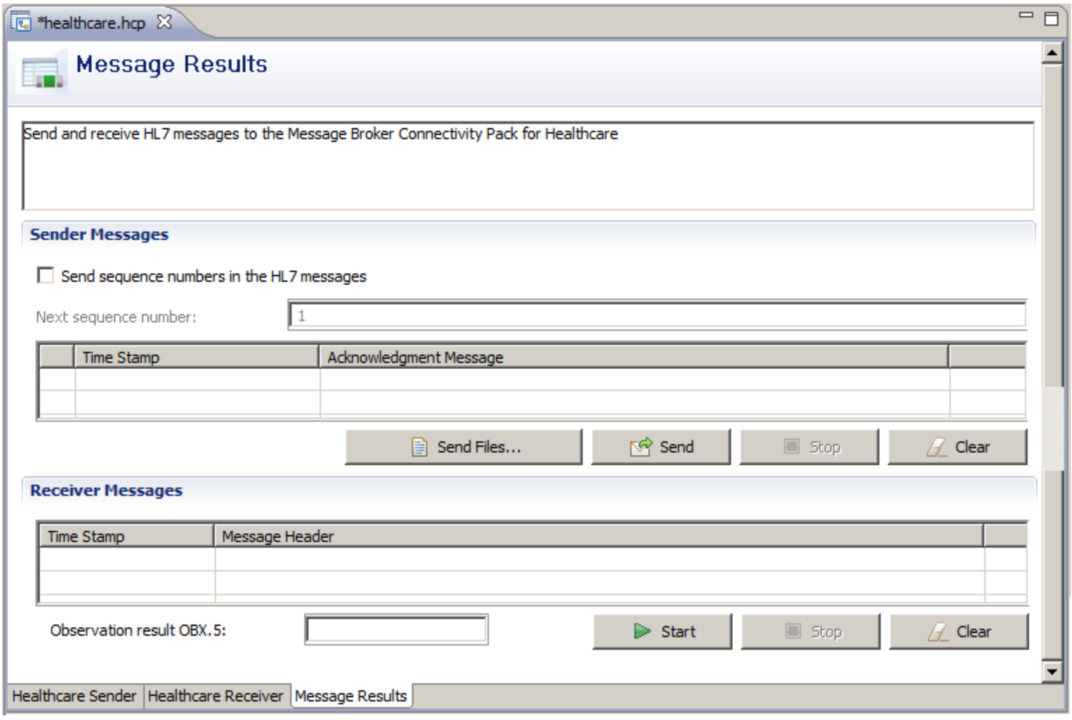
Leading bytes: 0B Trailing bytes: 1C0D

Acknowledgment Message

Acknowledgment type: Auto acknowledgment

File name:

5. Switch to the `Message Results` tab. This window controls the execution of the Healthcare test invocations.



Message Results

Send and receive HL7 messages to the Message Broker Connectivity Pack for Healthcare

Sender Messages

☐ Send sequence numbers in the HL7 messages

Next sequence number: 1

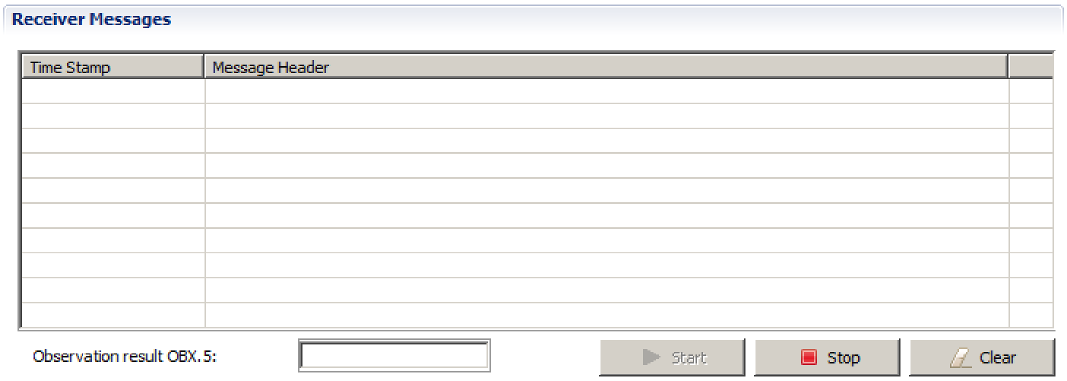
	Time Stamp	Acknowledgment Message

Receiver Messages

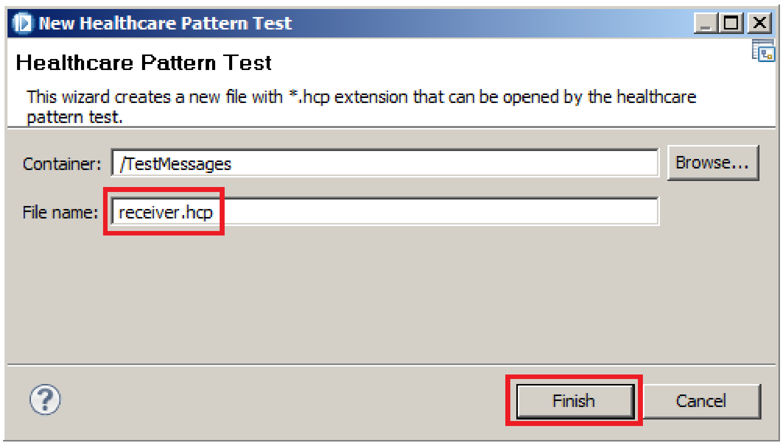
Time Stamp	Message Header

Observation result OBX.5:

6. First, click on `Start` (green arrow) to start the Receiving application. This will listen on the port specified on the Healthcare Receiver tab. Initially, no messages will be seen, but the Receiver can be seen to be running, since the `Stop` button has now become active.



7. The test tool's Receiver only stubs out one of the two destinations we require. In our scenario we would like to receive messages for both destinations at the same time, so now we will create a second Receiver. As before, from the File menu, choose `New`, `Healthcare Pattern Test`. Again, browse to the `TestMessages` container but this time specify a different filename such as `receiver.hcp`:



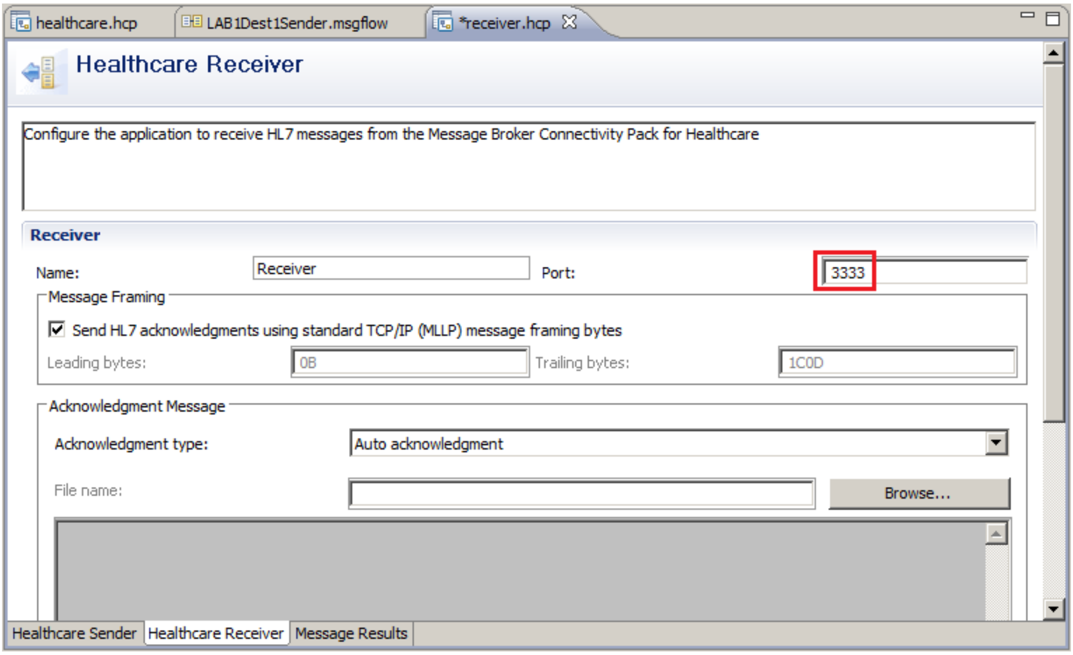
Healthcare Pattern Test

This wizard creates a new file with *.hcp extension that can be opened by the healthcare pattern test.

Container: /TestMessages

File name: receiver.hcp

8. In the resulting Healthcare Pattern Test file which is created, switch to the `Healthcare Receiver` tab and change the `Port` to be `3333`:



Configure the application to receive HL7 messages from the Message Broker Connectivity Pack for Healthcare

Receiver

Name: Receiver

Port: 3333

Message Framing

☑ Send HL7 acknowledgments using standard TCP/IP (MLLP) message framing bytes

Leading bytes: 0B

Trailing bytes: 1C0D

Acknowledgment Message

Acknowledgment type: Auto acknowledgment

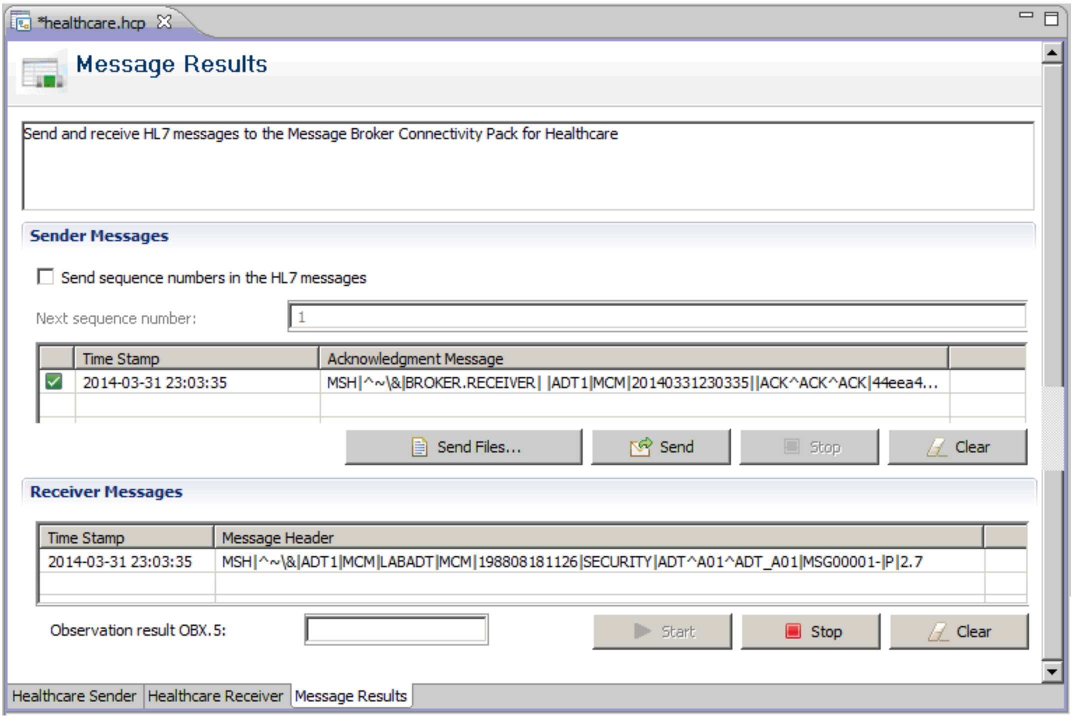
File name:

9. Switch to the `Message Results` tab and just as you did before, click on `Start` (green arrow) to start the Receiving application. This will listen on the port 3333 specified on the Healthcare Receiver tab. Initially, no messages will be seen, but the Receiver can be seen to be running, since the `Stop` button will become active.

10. Now return back to the first test file, named `healthcare.hcp` and click the `Send` button. The A01 message specified on the Healthcare Sender tab will be sent to the HL7 message flow.

    On the screen below, you can see the message has been sent to the HL7 message flow, has been processed, and has been received by the receiving application. Although not shown here, you should also be able to switch to the other test file, named receiver.hcp, and view the message sent to the other output destinations.

    If when you are testing yourself (for example using the built in IIB visual flow debugger) in some circumstances you may find messages end up in the Lab1.ERR queue. This will most commonly be because the standard MLLP protocol defines specific timeout and retry values. Since some messages are not being delivered in a timely fashion, they are eventually written to the Error queue.



Message Results

Send and receive HL7 messages to the Message Broker Connectivity Pack for Healthcare

Sender Messages

□ Send sequence numbers in the HL7 messages

Next sequence number: 1

	Time Stamp	Acknowledgment Message
☑	2014-03-31 23:03:35	MSH|^~\&|BROKER.RECEIVER| |ADT1|MCM|20140331230335||ACK^ACK^ACK|44eea4...

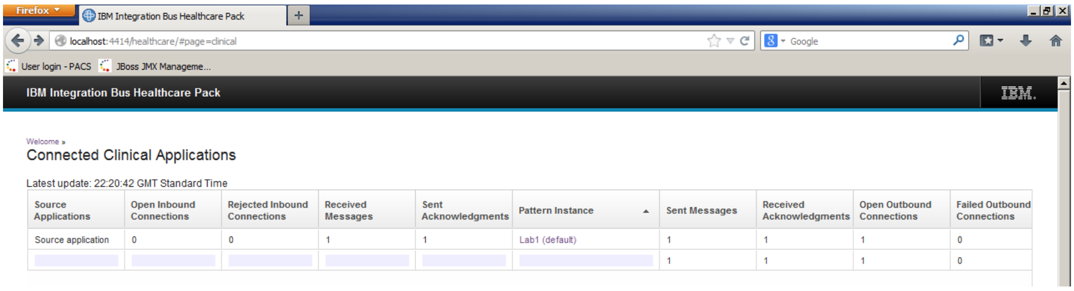
Receiver Messages

Time Stamp	Message Header
2014-03-31 23:03:35	MSH|^~\&|ADT1|MCM|LABADT|MCM|198808181126|SECURITY|ADT^A01^ADT_A01|MSG00001-|P|2.7

Observation result OBX.5:

11. Finally, ensure the Receivers of both test files are stopped, in order to avoid interference with any later work you choose to do.

12. If you return to the open page in your Web Browser session, you should see that the table now has numbers reflecting the successful delivery of messages between Source and destinations:



IBM Integration Bus Healthcare Pack

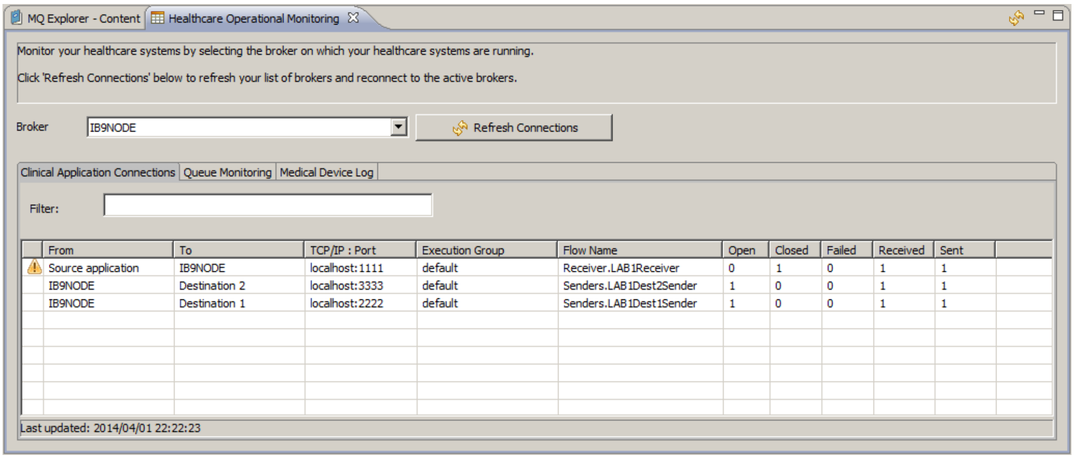
Welcome »

Connected Clinical Applications

Latest update: 22:20:42 GMT Standard Time

Source Applications	Open Inbound Connections	Rejected Inbound Connections	Received Messages	Sent Acknowledgments	Pattern Instance	Sent Messages	Received Acknowledgments	Open Outbound Connections	Failed Outbound Connections
Source application	0	0	1	1	Lab1 (default)	1	1	1	0
						1	1	1	0

Likewise, the Integration Explorer views should also reflect the test activity:



Monitor your healthcare systems by selecting the broker on which your healthcare systems are running.

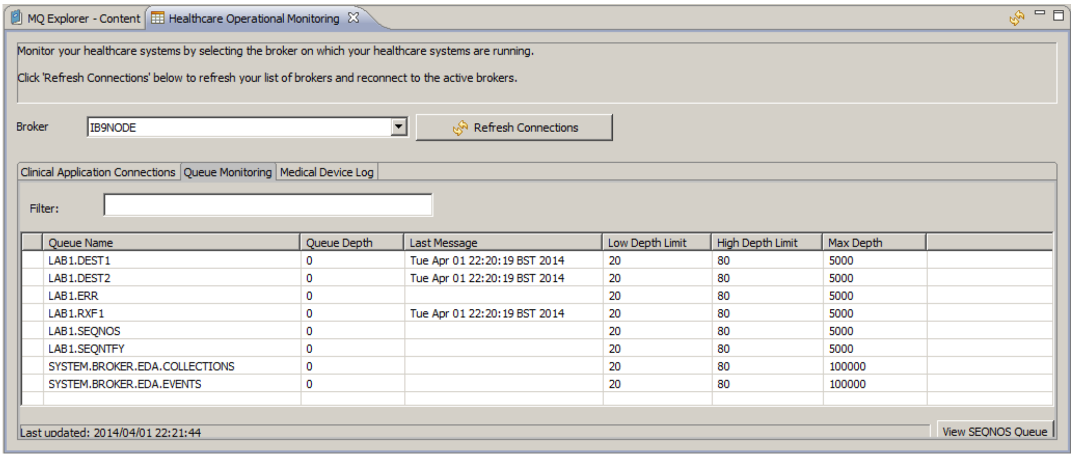
Click 'Refresh Connections' below to refresh your list of brokers and reconnect to the active brokers.

Broker: IB9NODE

Filter:

From	To	TCP/IP : Port	Execution Group	Flow Name	Open	Closed	Failed	Received	Sent
⚠ Source application	IB9NODE	localhost:1111	default	Receiver.LAB1Receiver	0	1	0	1	1
IB9NODE	Destination 2	localhost:3333	default	Senders.LAB1Dest2Sender	1	0	0	1	1
IB9NODE	Destination 1	localhost:2222	default	Senders.LAB1Dest1Sender	1	0	0	1	1

Last updated: 2014/04/01 22:22:23



Monitor your healthcare systems by selecting the broker on which your healthcare systems are running.

Click 'Refresh Connections' below to refresh your list of brokers and reconnect to the active brokers.

Broker: IB9NODE

Filter:

Queue Name	Queue Depth	Last Message	Low Depth Limit	High Depth Limit	Max Depth
LAB1.DEST1	0	Tue Apr 01 22:20:19 BST 2014	20	80	5000
LAB1.DEST2	0	Tue Apr 01 22:20:19 BST 2014	20	80	5000
LAB1.ERR	0		20	80	5000
LAB1.RXF1	0	Tue Apr 01 22:20:19 BST 2014	20	80	5000
LAB1.SEQNOS	0		20	80	5000
LAB1.SEQNTFY	0		20	80	5000
SYSTEM.BROKER.EDA.COLLECTIONS	0		20	80	100000
SYSTEM.BROKER.EDA.EVENTS	0		20	80	100000

Last updated: 2014/04/01 22:21:44

This concludes the Healthcare Integration Lab.