



IBM Integration Bus

Message Modeling with DFDL

Lab 6

Resolving Choices using Discriminators

June 2015

Hands-on lab built at product
Version 10.0.0.0



1. INTRODUCTION	3
1.1 LAB PREPARATION.....	3
1.2 LAB SCENARIO.....	4
2. BUILD THE MESSAGE MODEL	5
2.1 BUILD AND TEST THE MESSAGE MODEL WITHOUT DISCRIMINATORS	5
2.2 ADD THE DISCRIMINATORS TO THE MESSAGE MODEL	16
2.3 TEST THE MODEL WITH DISCRIMINATORS	20
END OF LAB GUIDE	28

1. Introduction

The DFDL standard provides a mechanism to allow a parser to make parsing decisions based on the content of other elements in a message. In this way, the structure and description of a message can be changed, and parsing of data can be optimized.

A DFDL parser is a recursive-descent parser with look-ahead used to resolve 'points of uncertainty', such as:

- A choice
- An optional element
- A variable array of elements

The DFDL parser must speculatively attempt to parse data until an object is either 'known to exist' or 'known not to exist'. Until that applies, the occurrence of a processing error causes the parser to suppress the error, back track and make another attempt.

The use of discriminators (the `dfdl:discriminator` annotation) can be used to assert that an object is 'known to exist', which prevents incorrect back-tracking. This lab will provide a simple example of the use of discriminators to illustrate this.

1.1 Lab preparation

To run this lab, unzip the supplied file `MessageModelling.zip` into the directory `c:\student` directory. This will create a subdirectory called `MessageModelling`, with several further subdirectories. If you are using the pre-supplied vmware image, this will already be available.

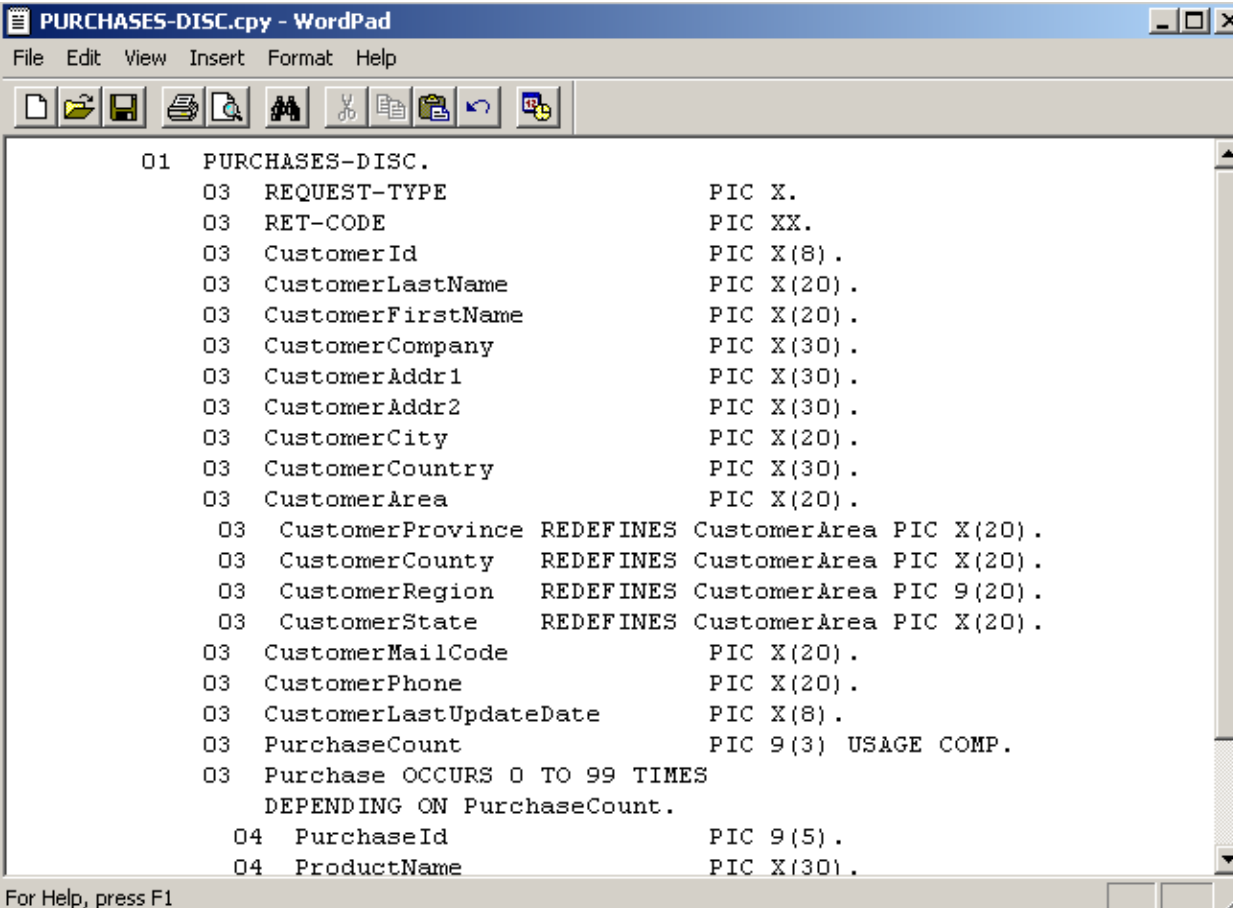
1.2 Lab Scenario

This lab will use an example based on a COBOL copybook which utilizes the REDEFINES clause. In a COBOL copybook, a REDEFINES clause enables a single element to contain different types of data (for example character or binary), and for a receiving application to process the element differently, depending on the type of data contained in the element.

The following COBOL copybook will be used in this lab. The key point to note is the use of the REDEFINES clause. This is used four times, and redefines the base element CustomerArea.

1. The CustomerProvince redefines clause is used when the CustomerCountry = 'Canada'.
2. The CustomerCounty redefines clause is used when the CustomerCountry = 'UK' or 'Ireland'.
3. The CustomerRegion redefines clause is used when the CustomerCountry = 'Russia'. Note that the Russian region is defined as PIC 9 (ie. a numeric value), even though the base element (CustomerState) is PIC X (character).
4. CustomerState is used when the CustomerCountry = 'USA'
5. CustomerArea will be used by any other country.

This copybook is supported by several data files, each containing a single record, with data corresponding to the above definitions.



```

01 PURCHASES-DISC.
   03 REQUEST-TYPE          PIC X.
   03 RET-CODE              PIC XX.
   03 CustomerId           PIC X(8) .
   03 CustomerLastName     PIC X(20) .
   03 CustomerFirstName    PIC X(20) .
   03 CustomerCompany      PIC X(30) .
   03 CustomerAddr1       PIC X(30) .
   03 CustomerAddr2       PIC X(30) .
   03 CustomerCity        PIC X(20) .
   03 CustomerCountry     PIC X(30) .
   03 CustomerArea        PIC X(20) .
      03 CustomerProvince REDEFINES CustomerArea PIC X(20) .
      03 CustomerCounty  REDEFINES CustomerArea PIC X(20) .
      03 CustomerRegion  REDEFINES CustomerArea PIC 9(20) .
      03 CustomerState   REDEFINES CustomerArea PIC X(20) .
   03 CustomerMailCode    PIC X(20) .
   03 CustomerPhone       PIC X(20) .
   03 CustomerLastUpdateDate PIC X(8) .
   03 PurchaseCount       PIC 9(3) USAGE COMP.
   03 Purchase OCCURS 0 TO 99 TIMES
      DEPENDING ON PurchaseCount.
   04 PurchaseId          PIC 9(5) .
   04 ProductName        PIC X(30) .

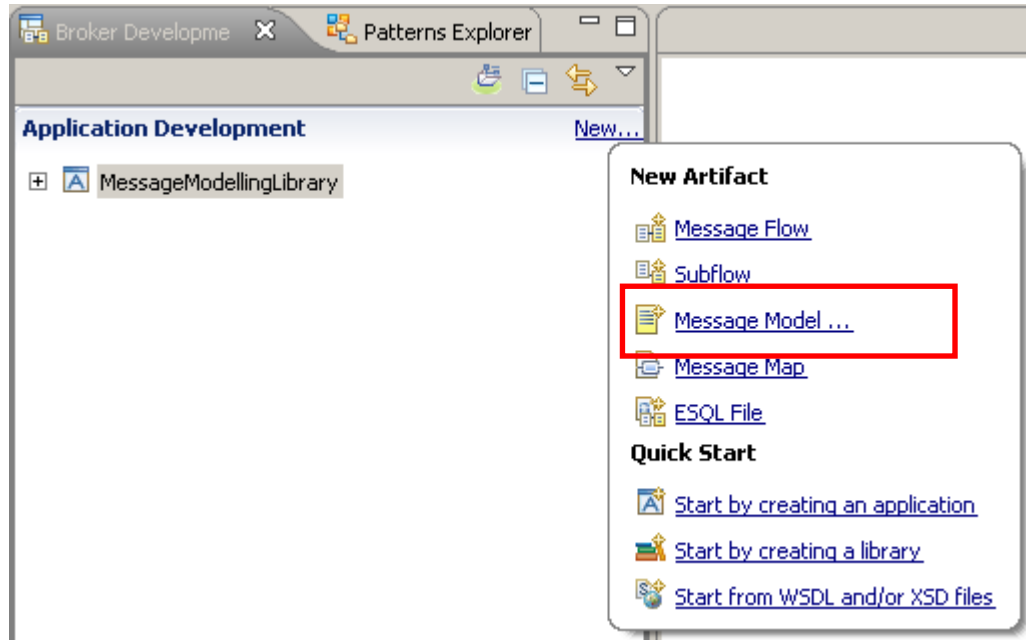
```

For Help, press F1

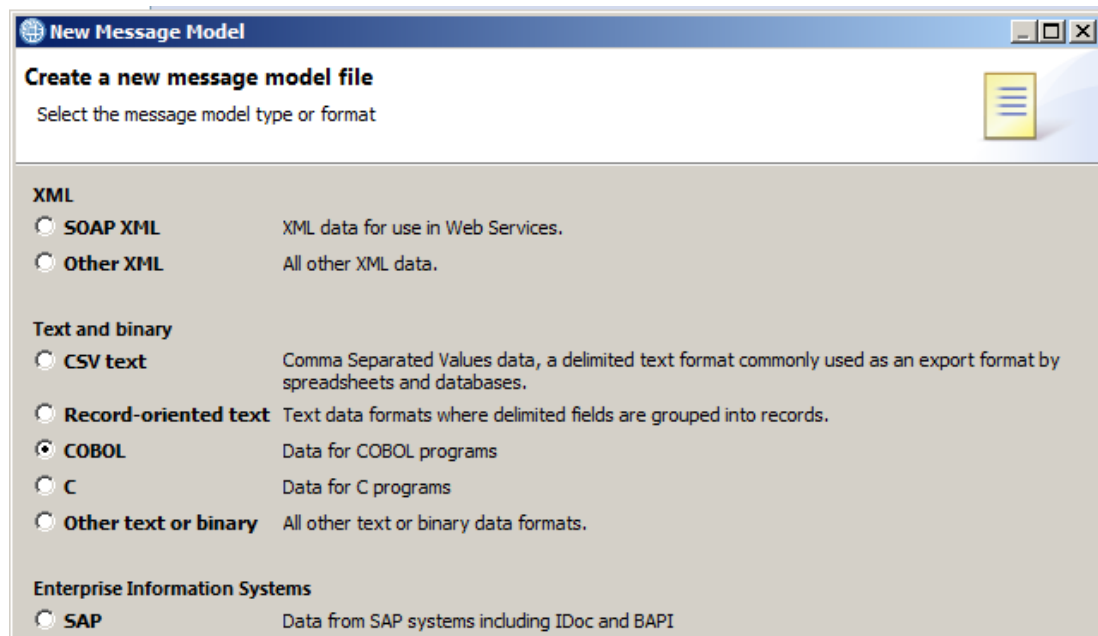
2. Build the Message Model

2.1 Build and test the Message Model without Discriminators

1. In the MessageModellingLibrary that you created in Lab1, click New -> Message Model (or create a new library for the purpose of this lab).

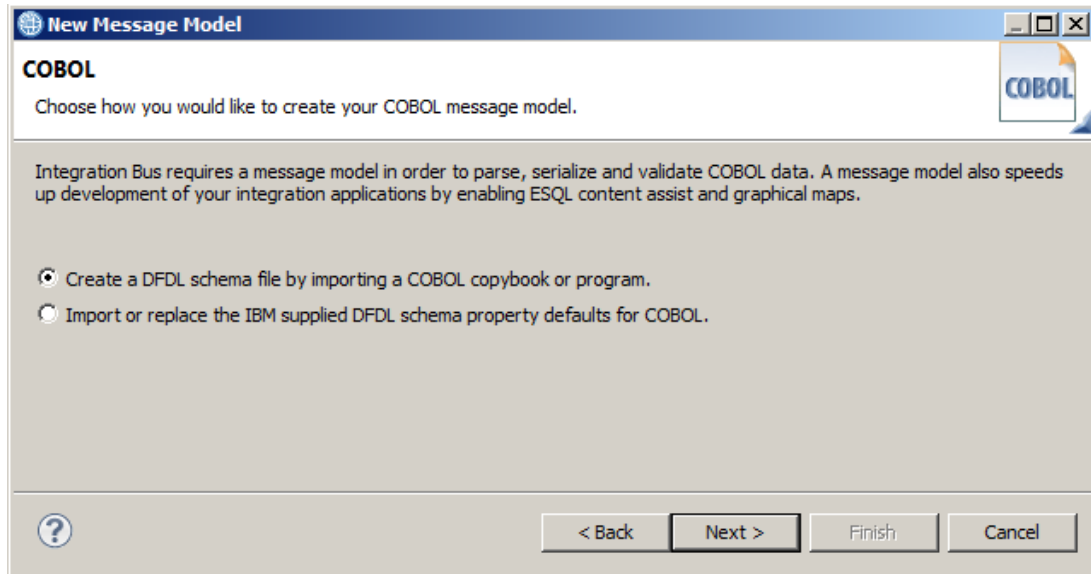


2. In the "New Message Model" window, select "COBOL" and click Next.



3. You can create the new message model using a wizard or create an empty DFDL schema and start from scratch.

Leave the default selection to “Create a DFDL schema by importing a COBOL copybook” and click Next.



- Click "Select source from outside workspace", and use Browse to navigate to c:\student10\messagemodelling\discriminators.

Select the PURCHASES-DISC.cpy file and click Open, then Next.

New DFDL Schema

New Data Format Description Language Schema (DFDL) From a Source

Select the type of data definition from which to import and create a DFDL schema.

Application or Library: MessageModellingLibrary

Folder: <Specifying a folder is optional>

DFDL schema file name: PURCHASES-DISC.xsd

Select source file from workspace:

MessageModellingLibrary

Select source file from outside workspace:

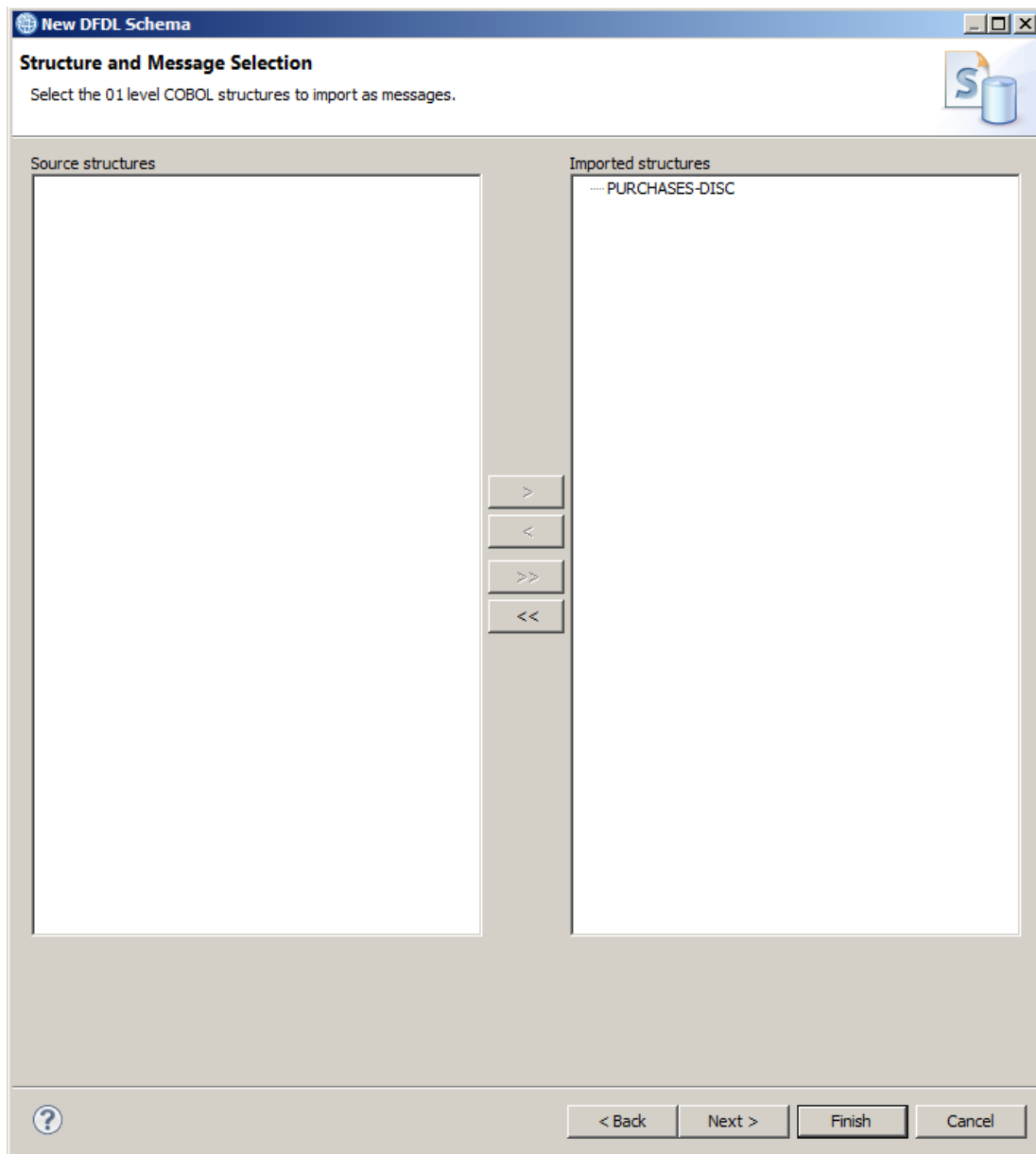
Location: C:\student10\MessageModeling\discriminators\PURCHASES-DISC.cpy

Overwrite existing DFDL schema file

Copy source file into the 'importFiles' directory of the target project

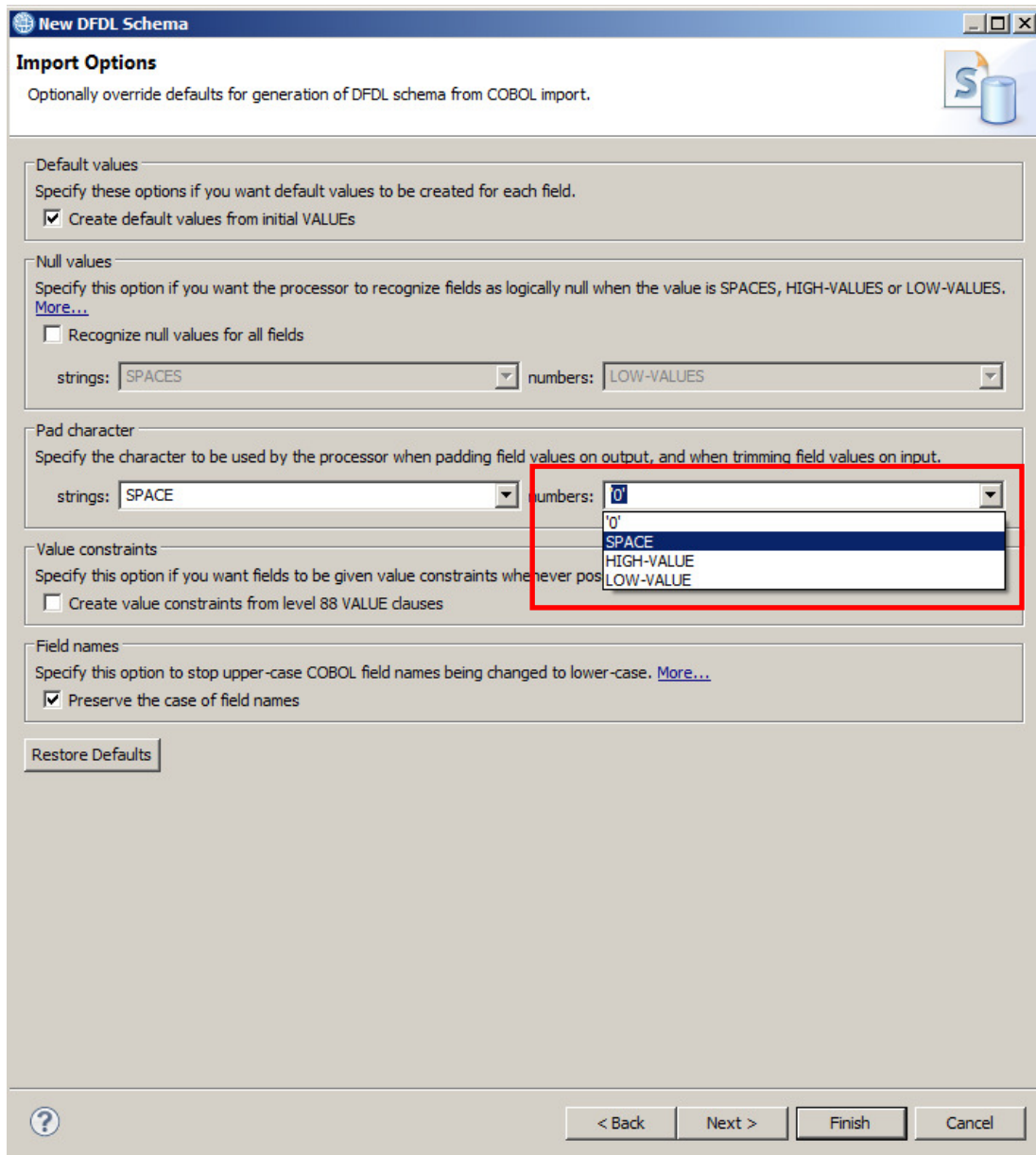
5. Select the PURCHASES-DISC structure and move it to the right side of the window (use the arrows in the centre of the window).

Click Next (**do NOT** click Finish).



- In the “Pad Character” definition, for the “numbers” padding, select SPACE from the drop-down menu.

Click Next.



New DFDL Schema

Import Options
Optionally override defaults for generation of DFDL schema from COBOL import.

Default values
Specify these options if you want default values to be created for each field.
 Create default values from initial VALUES

Null values
Specify this option if you want the processor to recognize fields as logically null when the value is SPACES, HIGH-VALUES or LOW-VALUES.
[More...](#)
 Recognize null values for all fields
strings: SPACES numbers: LOW-VALUES

Pad character
Specify the character to be used by the processor when padding field values on output, and when trimming field values on input.
strings: SPACE numbers: 0
SPACE
HIGH-VALUE
LOW-VALUE

Value constraints
Specify this option if you want fields to be given value constraints whenever possible.
 Create value constraints from level 88 VALUE clauses

Field names
Specify this option to stop upper-case COBOL field names being changed to lower-case. [More...](#)
 Preserve the case of field names

Restore Defaults

? < Back Next > Finish Cancel

7. Click Finish.

New DFDL Schema

Import Options

Specify settings that describe the COBOL data as it appears on the target system.

The importer compiles the supplied COBOL file in order to create the DFDL schema. Some compiler options may be changed on this page. A full set of compiler options may be found in the Preferences settings [here](#).

Specifying target platform affects selection of the encoding options.

Platform: Win32

Encoding options

Dynamic - provided to the processor by the application at runtime

Fixed - provided below

Encoding (code page): ISO-8859-1

Floating point format: IEEE Non-Extended

Byte order: Little endian Big endian

External (zoned) decimal

Use EBCDIC sign characters with ASCII numbers

EBCDIC encoding (code page) for sign characters:

Compiler options

QUOTE: DOUBLE SINGLE

TRUNC: STD OPT BIN

NSYMBOL: NATIONAL DBCS

Restore Defaults

? < Back Next > Finish Cancel

- When the wizard finishes, the DFDL Editor will open with the generated PURCHASESDISC.xsd schema file.

You will see that a “local choice” has been defined in the model, based on the COBOL REDEFINES clauses. Expanding the “choice” element, you will see that the first choice element is CustomerArea, which corresponds to the primary definition of this item in the copybook.

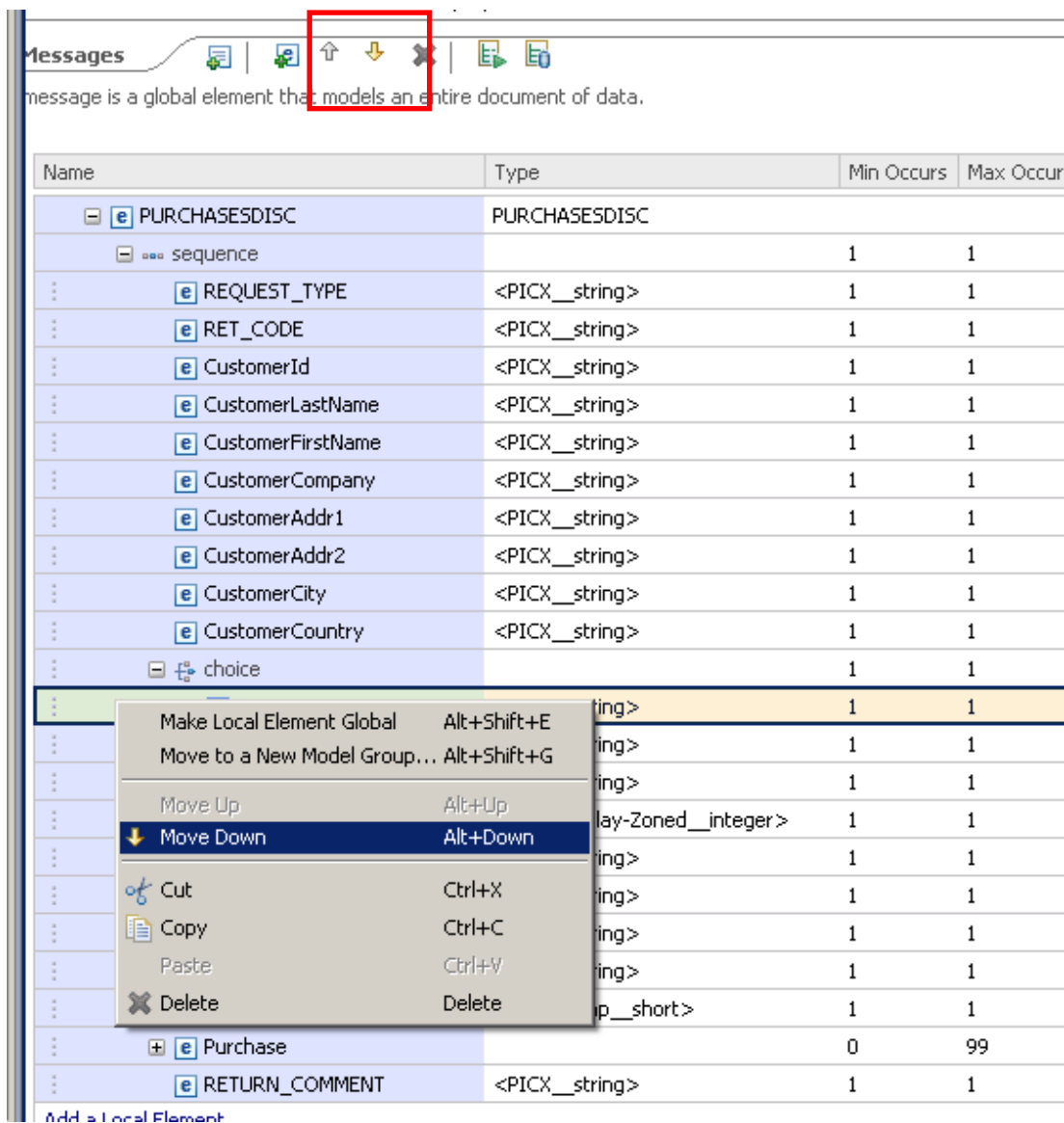
The remaining choice elements correspond to the COBOL items in the copybook with the REDEFINES keyword. The order that the choice elements appear is determined by the order of the elements in the COBOL copybook, so the first such element is CustomerArea.

In this scenario, we want to treat this choice CustomerArea as the default. Since choice routes are evaluated in order, we will move this choice route to the bottom of the choices.

Name	Type	Min Occurs	Max Occurs	Default Value
[-] e PURCHASESDISC	PURCHASESDISC			
[-] sequence		1	1	
... e REQUEST_TYPE	<PICX_string>	1	1	
... e RET_CODE	<PICX_string>	1	1	
... e CustomerId	<PICX_string>	1	1	
... e CustomerLastName	<PICX_string>	1	1	
... e CustomerFirstName	<PICX_string>	1	1	
... e CustomerCompany	<PICX_string>	1	1	
... e CustomerAddr1	<PICX_string>	1	1	
... e CustomerAddr2	<PICX_string>	1	1	
... e CustomerCity	<PICX_string>	1	1	
... e CustomerCountry	<PICX_string>	1	1	
... choice		1	1	
... e CustomerArea	<PICX_string>	1	1	
... e CustomerProvince	<PICX_string>	1	1	
... e CustomerCounty	<PICX_string>	1	1	
... e CustomerRegion	<PIC9-Display-Zoned_integer>	1	1	0
... e CustomerState	<PICX_string>	1	1	
... e CustomerMailCode	<PICX_string>	1	1	
... e CustomerPhone	<PICX_string>	1	1	
... e CustomerLastUpdateDate	<PICX_string>	1	1	
... e PurchaseCount	<PIC9-Comp_short>	1	1	0
... [-] Purchase		0	99	
... e RETURN_COMMENT	<PICX_string>	1	1	

[Add a Local Element](#)

- Right-click the element CustomerArea (to the left of the element name), and select “Move Down”. Alternatively, you can highlight the element name and click the yellow down-arrow, as shown below.



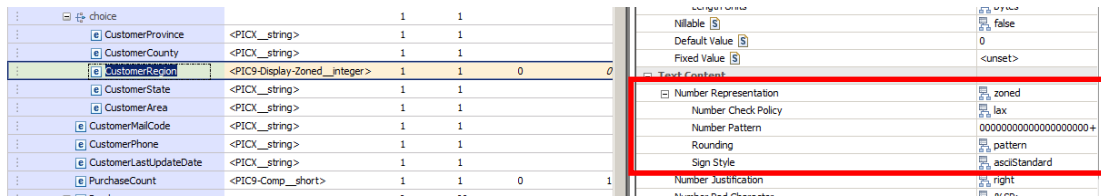
- Move the CustomerArea element down to the bottom of the choice item. The final result should look like this:

[-] choice		1	1	
... e CustomerProvince	<PICX_string>	1	1	
... e CustomerCounty	<PICX_string>	1	1	
... e CustomerRegion	<PIC9-Display-Zoned_integer>	1	1	0
... e CustomerState	<PICX_string>	1	1	
... e CustomerArea	<PICX_string>	1	1	
... e CustomerMailCode	<PICX_string>	1	1	

- Note that most of the choice elements are string elements. However, the CustomerRegion element is defined as PIC9-Display-Zoned_integer.

Highlight the CustomerRegion element.

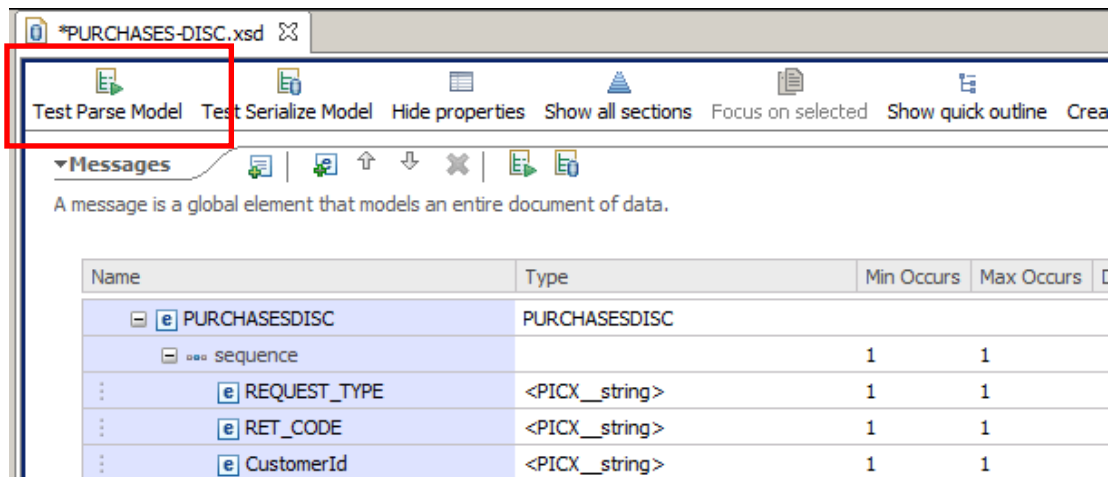
In the Representation Properties, you will see that the Text Number Representation section has had several properties set to reflect the nature of the numeric data, and the fact that we want any leading blank characters (“spaces” in COBOL parlance) to be removed from the parsed data.



- We will now test the base model by using the Test Parse tool.

Save the model.

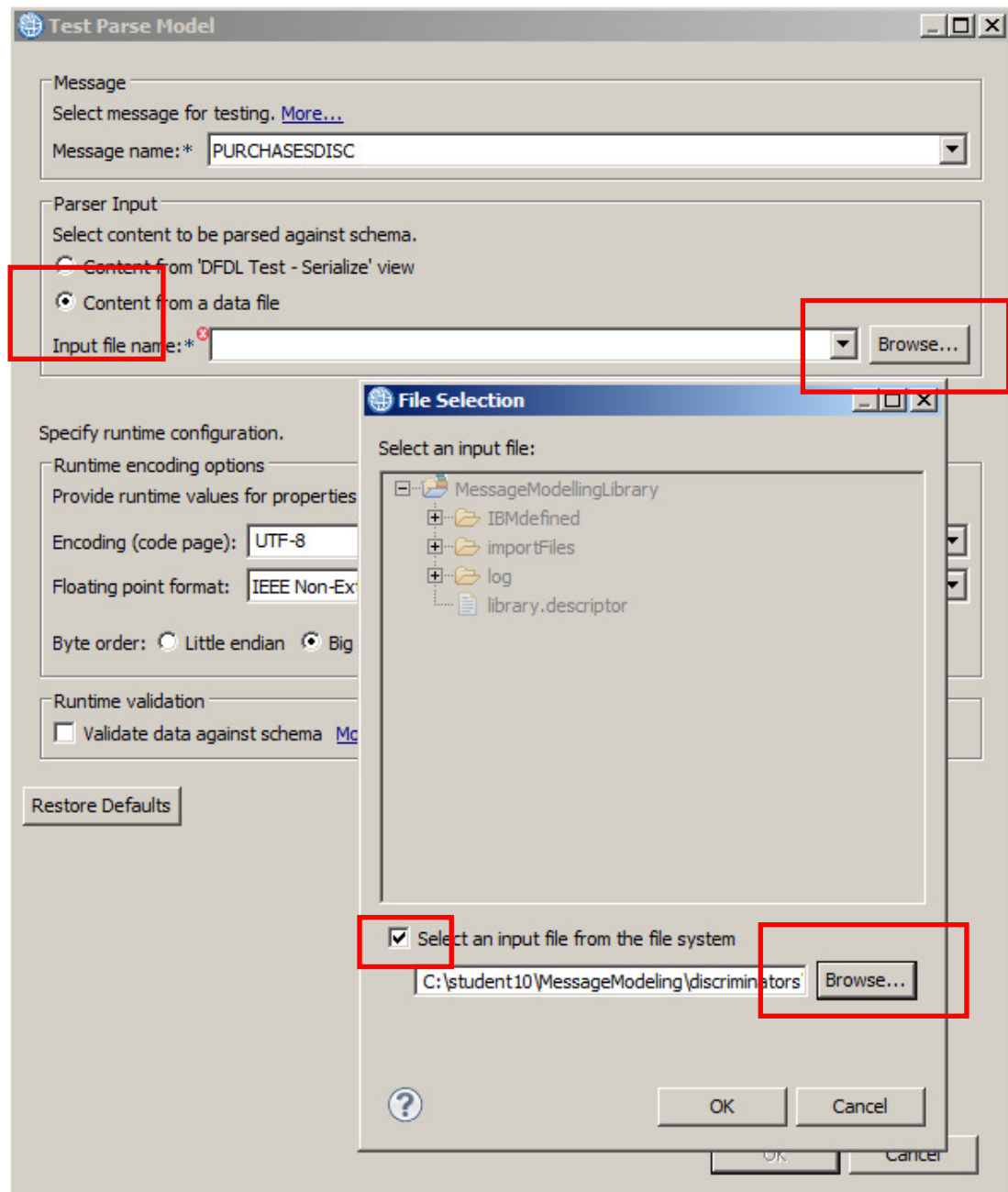
Click Test Parse Model.



13. Select "Content from a data file", then Browse, then tick "Select an input file from the file system".

Using the Browse button, navigate to c:\student\messagemodelling\discriminators, and select the file Purchases_disc_USA.dat.

Click OK, then OK and then confirm the switch to the DFDL Test perspective.



14. The Test Parse will run. In the DFDL Test Logical Instance view, you will see that the data has been fully parsed. Note that the CustomerCountry is USA, but the value of "Texas" has been placed into the choice element called CustomerProvince.

The reason that CustomerProvince has been selected is that the branches of the choice are tried in the declared order until one parses successfully. Because CustomerState and CustomerProvince are both declared as PIC X(30), they end up with the same properties in the schema. The result is that when USA data is parsed, the CustomerArea element will always match the 'CustomerProvince' data.

This is clearly not quite what we need, so we will need to refine the message model with some discriminators.

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

Message: PURCHASESDISC (/student/workspace/MessageModellingLibrary/PURCHASES-DISC.xsd)

Tree View XML View

Name	Type	Value
[-] PURCHASESDISC		<input type="checkbox"/>
REQUEST_TYPE	xs:string	A
RET_CODE	xs:string	00
CustomerId	xs:string	12345678
CustomerLastName	xs:string	Griffin
CustomerFirstName	xs:string	Peter
CustomerCompany	xs:string	Pawtucket Brew...
CustomerAddr1	xs:string	31 Spooner st.
CustomerAddr2	xs:string	456 1st av.
CustomerCity	xs:string	Quabog
CustomerCountry	xs:string	USA
CustomerProvince	xs:string	Texas
CustomerMailCode	xs:string	12312
CustomerPhone	xs:string	123-123-1234
CustomerLastUpdateC	xs:string	04082008
PurchaseCount	xs:unsignedShort	4
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
RETURN_COMMENT	xs:string	none

2.2 Add the Discriminators to the Message Model

In this section, you will now add discriminators to several of the elements in the message model. This will allow the parser to dynamically select the appropriate elements, based on values in the incoming message. The parser still parses branches in the declared order until one is found that parses successfully, but the discriminators mean that the data now only matches one of the branches.

This will be done by setting a discriminator on the CustomerProvince, CustomerCounty, CustomerRegion and CustomerState elements. No change will be made to the CustomerArea element, which will act as a default for all countries that do not have explicit discriminators.

1. Switch back to the Application Development perspective.

In the Message Model editor, expand the “choice” element, and click on the CustomerProvince element, and then click the “Asserts and Discriminators” tab on the right side.

Click the Discriminator radio button.

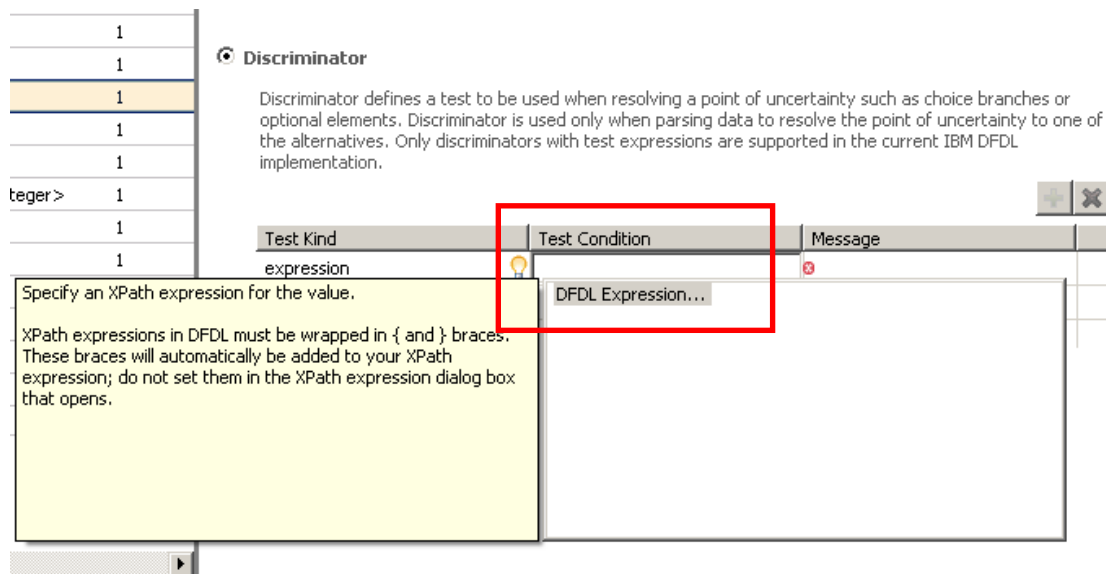
Note that there is no discriminator for this element at the moment.

The screenshot shows the IBM Message Model editor interface. On the left, a tree view displays the message model structure, including a 'choice' element containing 'CustomerProvince'. The 'CustomerProvince' element is selected. On the right, the 'Asserts and Discriminators' tab is active. The 'Discriminator' radio button is selected. Below it, a table with columns 'Test Kind', 'Test Condition', and 'Message' is shown. The 'Add discriminator' link is highlighted with a red box. Another red box highlights the 'Asserts and Discriminators' tab label at the top of the right pane.

2. Click Add Discriminator. This will allow you to edit the “Test Condition” field.

This is a close-up of the 'Discriminator' configuration dialog. It contains a text area with the following text: "Discriminator defines a test to be used when resolving a point of uncertainty such as choice branches or optional elements. Discriminator is used only when parsing data to resolve the point of uncertainty to one of the alternatives. Only discriminators with test expressions are supported in the current IBM DF DL implementation." Below the text area is a table with columns 'Test Kind', 'Test Condition', and 'Message'. The 'Test Condition' field is highlighted with a red box, and the 'Add discriminator' link is also highlighted with a red box.

- The easiest way to populate this field is to use the Content Assist function. With the mouse in the Test Condition field, use the key sequence "Control-space", then immediately use the Return key (or double-click "DFDL Expression").



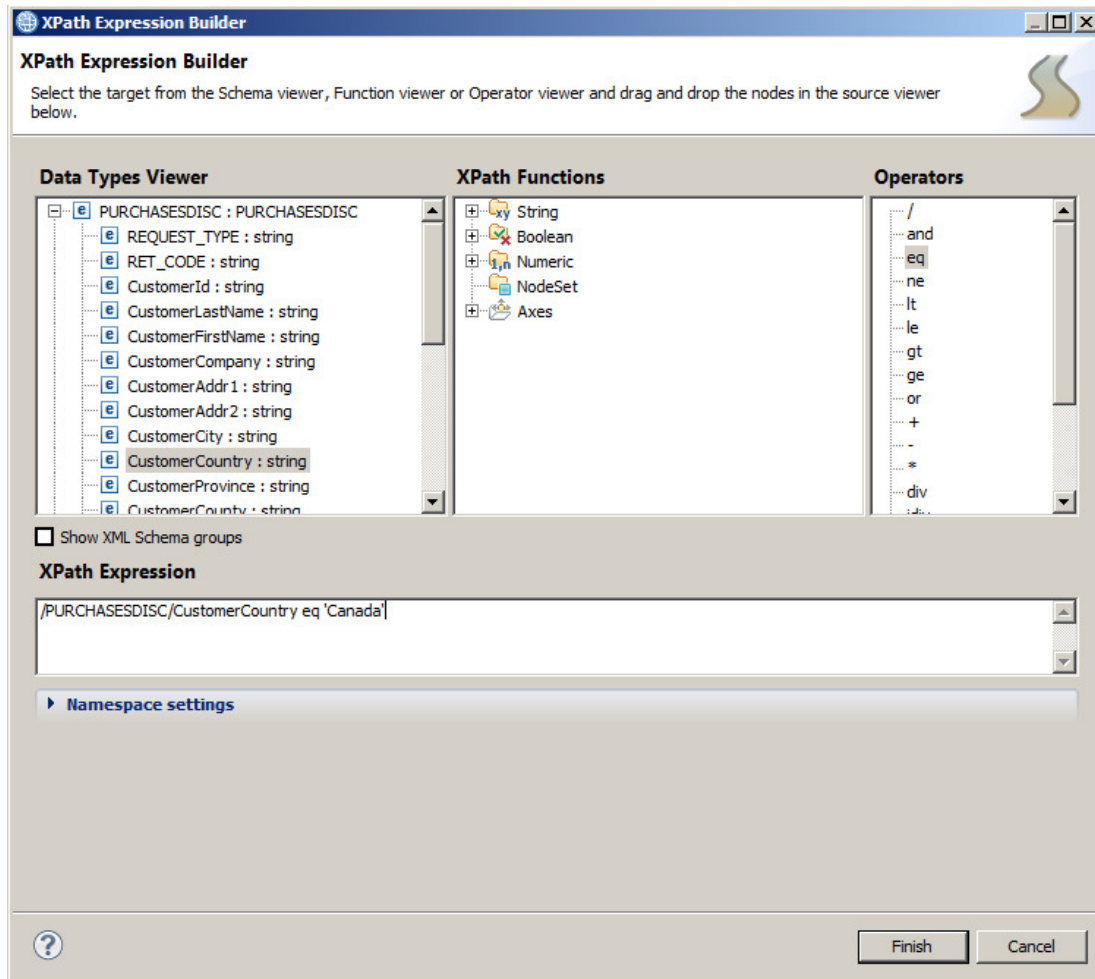
- This will open the XPath Expression Builder.

We will generate an XPath expression that will check the CustomerCountry element for the value "Canada". When this value is detected, the parser will use the element CustomerProvince to parse the data in the CustomerState element (redefined by the element CustomerProvince).

Expand the Data Types Viewer, and drag the CustomerCountry element to the XPath expression pane. Complete the XPath expression manually. The final result should be:

```
/PURCHASESDISC/CustomerCountry eq 'Canada'
```

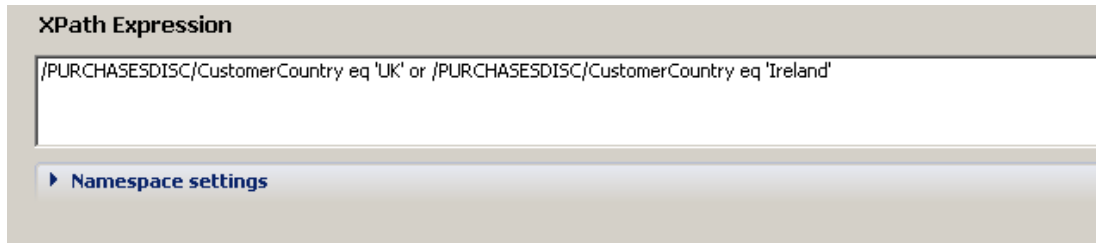
Click Finish.



5. Performs steps 1 – 4 for the element CustomerCountry. In this case, CustomerCountry should be used to parse the element if the CustomerCountry is “UK” or “Ireland”. (Do not confuse CustomerCountry with CustomerCounty).

The XPath expression for the element CustomerCountry should be:

```
/PURCHASESDISC/CustomerCountry eq 'UK' or  
/PURCHASESDISC/CustomerCountry eq 'Ireland'
```



XPath Expression

```
/PURCHASESDISC/CustomerCountry eq 'UK' or /PURCHASESDISC/CustomerCountry eq 'Ireland'
```

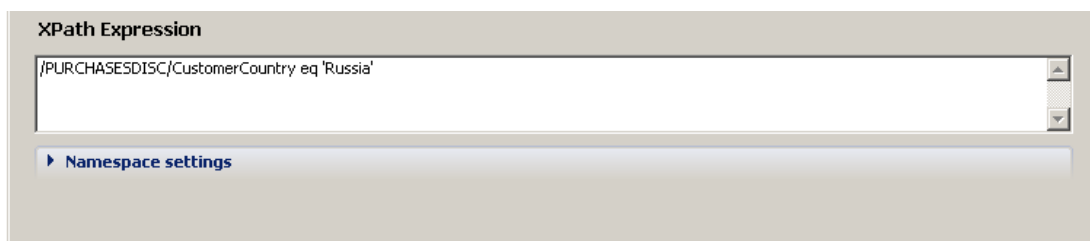
▶ **Namespace settings**

6. Performs steps 1 – 4 for the element CustomerRegion. In this case, CustomerRegion should be used to parse the element if the CustomerCountry is “Russia”.

The XPath expression for the element CustomerRegion should be:

```
/PURCHASESDISC/CustomerCountry eq 'Russia'
```

Remember that CustomerRegion is the element that will contain a numeric value, possibly with leading blanks.



XPath Expression

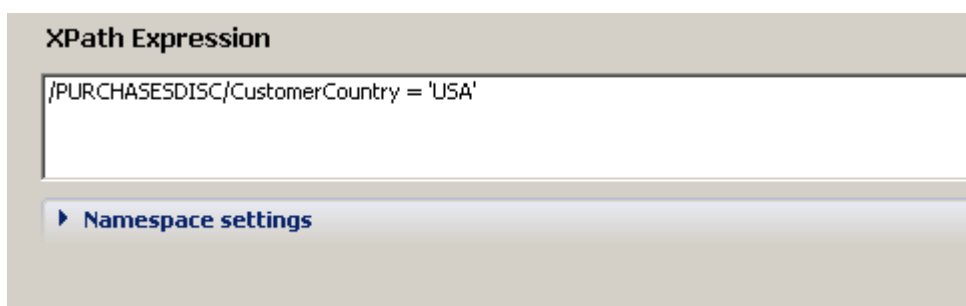
```
/PURCHASESDISC/CustomerCountry eq 'Russia'
```

▶ **Namespace settings**

7. Performs steps 1 – 4 for the element CustomerState. In this case, CustomerState should be used to parse the element if the CustomerCountry is “USA”.

The XPath expression for the element CustomerState should be:

```
/PURCHASESDISC/CustomerCountry eq 'USA'
```



XPath Expression

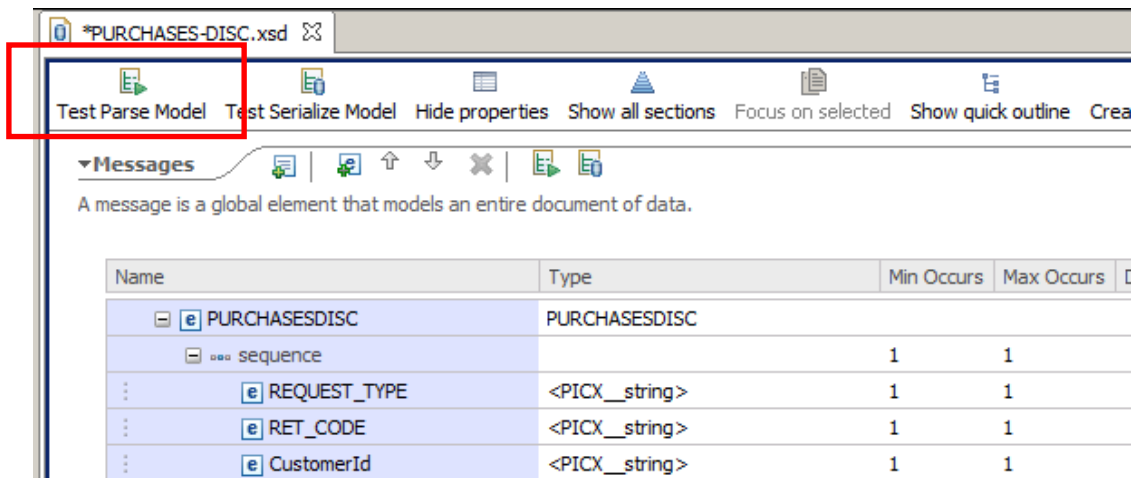
```
/PURCHASESDISC/CustomerCountry = 'USA'
```

▶ **Namespace settings**

8. Save the model

2.3 Test the model with discriminators

1. In the message model editor, click Test Parse Model.

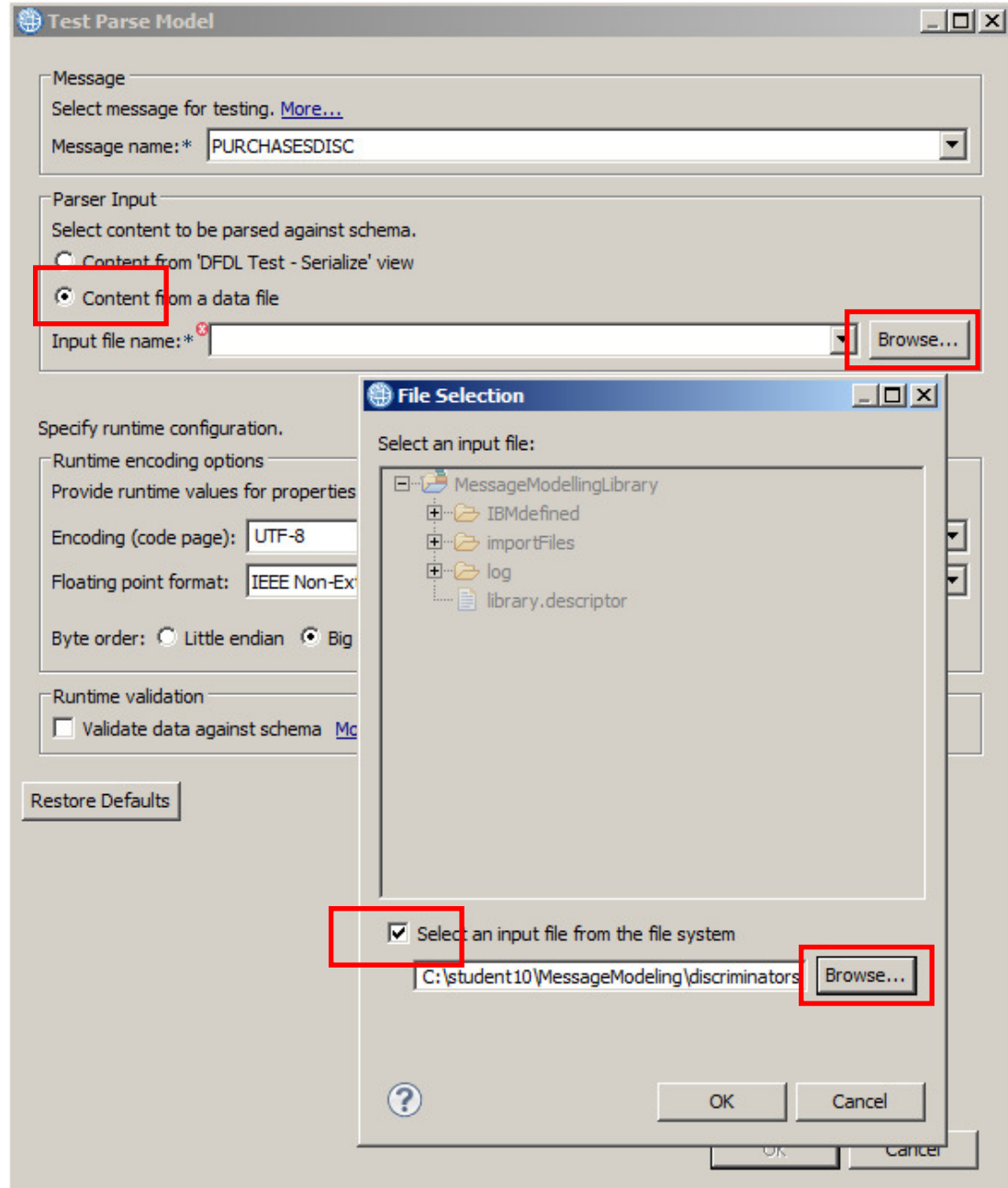


- As before, select "Content from a data file", click Browse, and select an input file from the file system.

Navigate to c:\student10\messagemodelling\discriminators.

Select the file Purchases_disc_USA.dat.

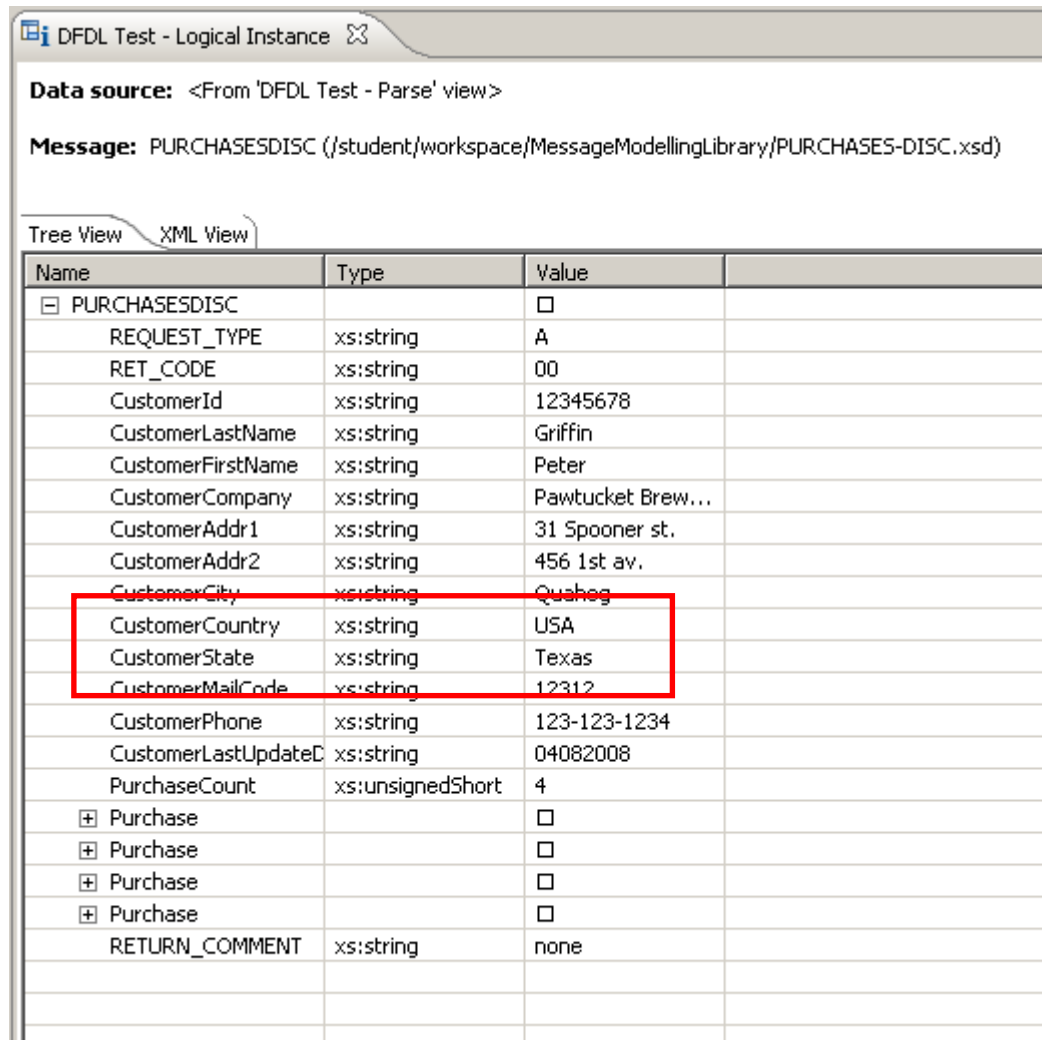
Click OK and OK. and then confirm the switch to the DFDL Test perspective.



- The file should parse fully.

In the DFDL Test Logical Instance view, you will see that the CustomerCountry is USA.

Because the discriminators have now been specified on the choice, the parser has detected that the value of “USA” matches a discriminator, and has therefore placed the value “Texas” into the element CustomerState, replacing the choice element CustomerProvince.



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

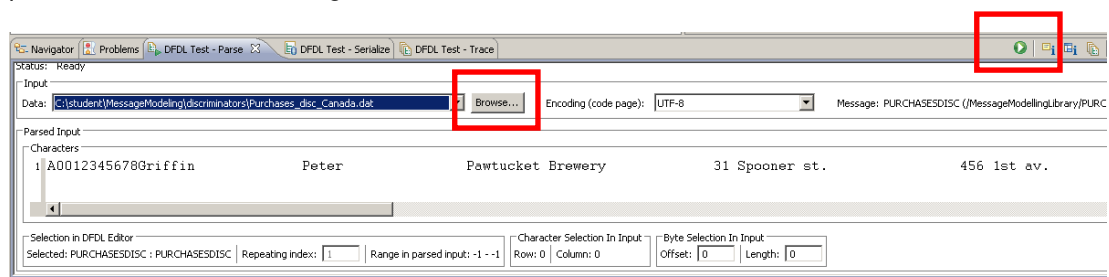
Message: PURCHASESDISC (/student/workspace/MessageModellingLibrary/PURCHASES-DISC.xsd)

Tree View XML View

Name	Type	Value
PURCHASESDISC		<input type="checkbox"/>
REQUEST_TYPE	xs:string	A
RET_CODE	xs:string	00
CustomerId	xs:string	12345678
CustomerLastName	xs:string	Griffin
CustomerFirstName	xs:string	Peter
CustomerCompany	xs:string	Pawtucket Brew...
CustomerAddr1	xs:string	31 Spooner st.
CustomerAddr2	xs:string	456 1st av.
CustomerCity	xs:string	Quahog
CustomerCountry	xs:string	USA
CustomerState	xs:string	Texas
CustomerMailCode	xs:string	12312
CustomerPhone	xs:string	123-123-1234
CustomerLastUpdateC	xs:string	04082008
PurchaseCount	xs:unsignedShort	4
Purchase		<input type="checkbox"/>
Purchase		<input type="checkbox"/>
Purchase		<input type="checkbox"/>
Purchase		<input type="checkbox"/>
RETURN_COMMENT	xs:string	none

This can be verified by looking at the trace entries in the DFDL Trace view.

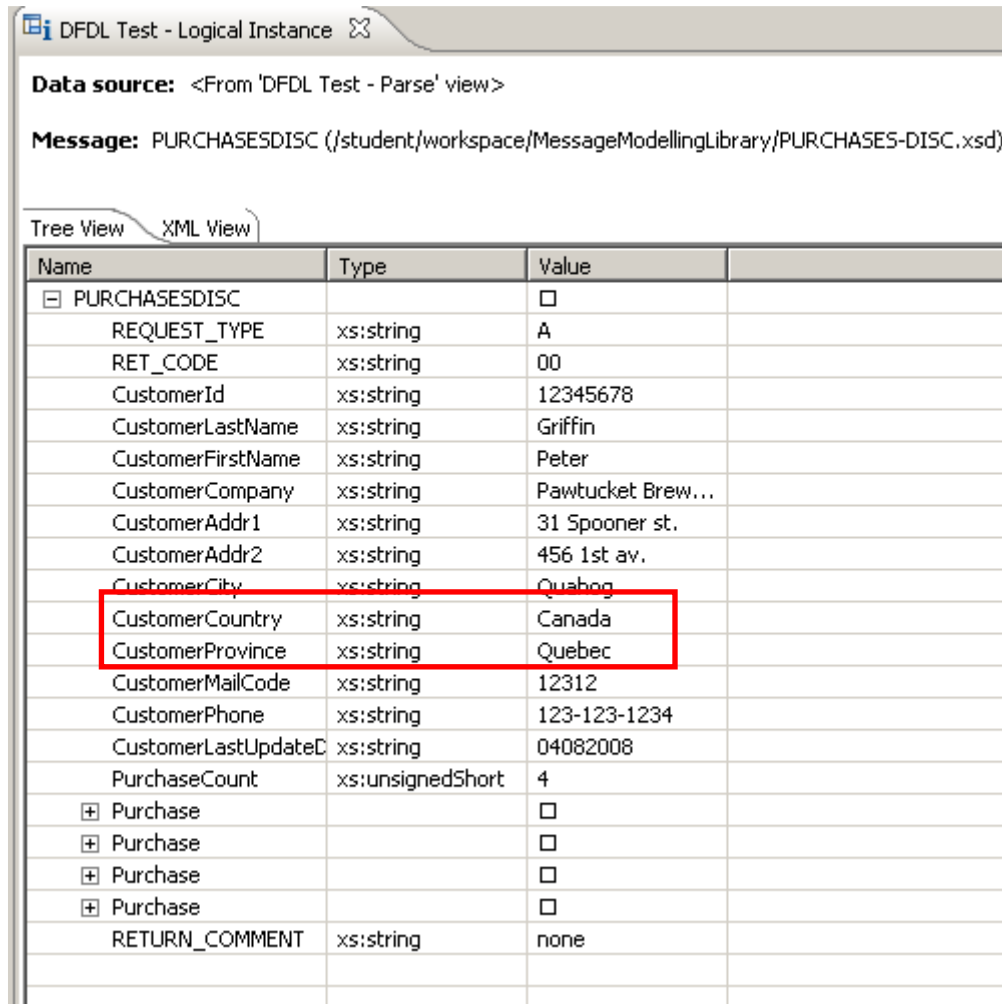
- Now select the input file Purchases_disc_Canada.dat, using the browse button in the bottom pane and then click on the green “Parse” button.



5. In the Logical Instance view, you will see that the CustomerCountry is “Canada”.

In this case, the parser has determined that the value in this element matches one of the specified discriminators, and has used the choice element CustomerProvince to represent the appropriate element, and has set the value to “Quebec” (shown highlighted below).

Note that the type of the CustomerProvince element is “string”.



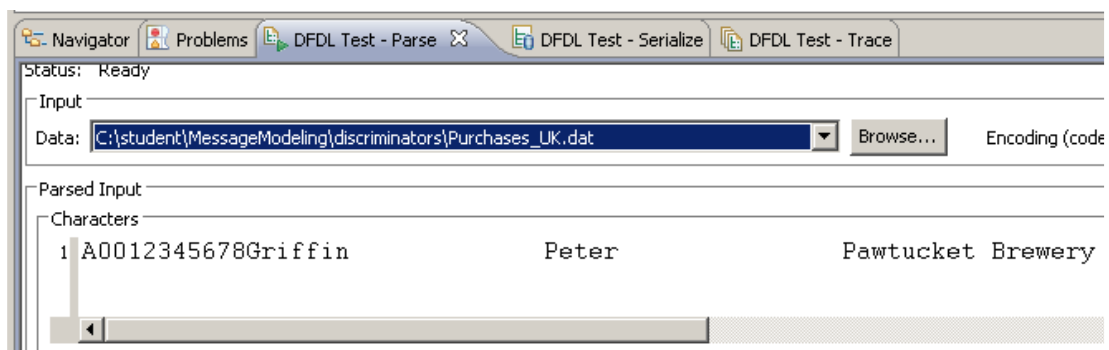
Data source: <From 'DFDL Test - Parse' view>

Message: PURCHASESDISC (/student/workspace/MessageModellingLibrary/PURCHASES-DISC.xsd)

Tree View XML View

Name	Type	Value
[-] PURCHASESDISC		<input type="checkbox"/>
REQUEST_TYPE	xs:string	A
RET_CODE	xs:string	00
CustomerId	xs:string	12345678
CustomerLastName	xs:string	Griffin
CustomerFirstName	xs:string	Peter
CustomerCompany	xs:string	Pawtucket Brew...
CustomerAddr1	xs:string	31 Spooner st.
CustomerAddr2	xs:string	456 1st av.
CustomerCity	xs:string	Quabog
CustomerCountry	xs:string	Canada
CustomerProvince	xs:string	Quebec
CustomerMailCode	xs:string	12312
CustomerPhone	xs:string	123-123-1234
CustomerLastUpdateC	xs:string	04082008
PurchaseCount	xs:unsignedShort	4
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
RETURN_COMMENT	xs:string	none

6. Now select the input file Purchases_disc_UK.dat, and click on the green “Parse” button.



7. In the Logical Instance view, you will see that the CustomerCountry is “UK”.

In this case, the parser has determined that the value in this element matches one of the specified discriminators, and has used the element CustomerCountry to parse the appropriate element. It has replaced the element CustomerState with the element CustomerCounty, and has set the value to “Hampshire” (shown highlighted below).

DFDL Test - Logical Instance

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

Message: PURCHASESDISC (/student/workspace/MessageModellingLibrary/PURCHASES-DISC.xsd)

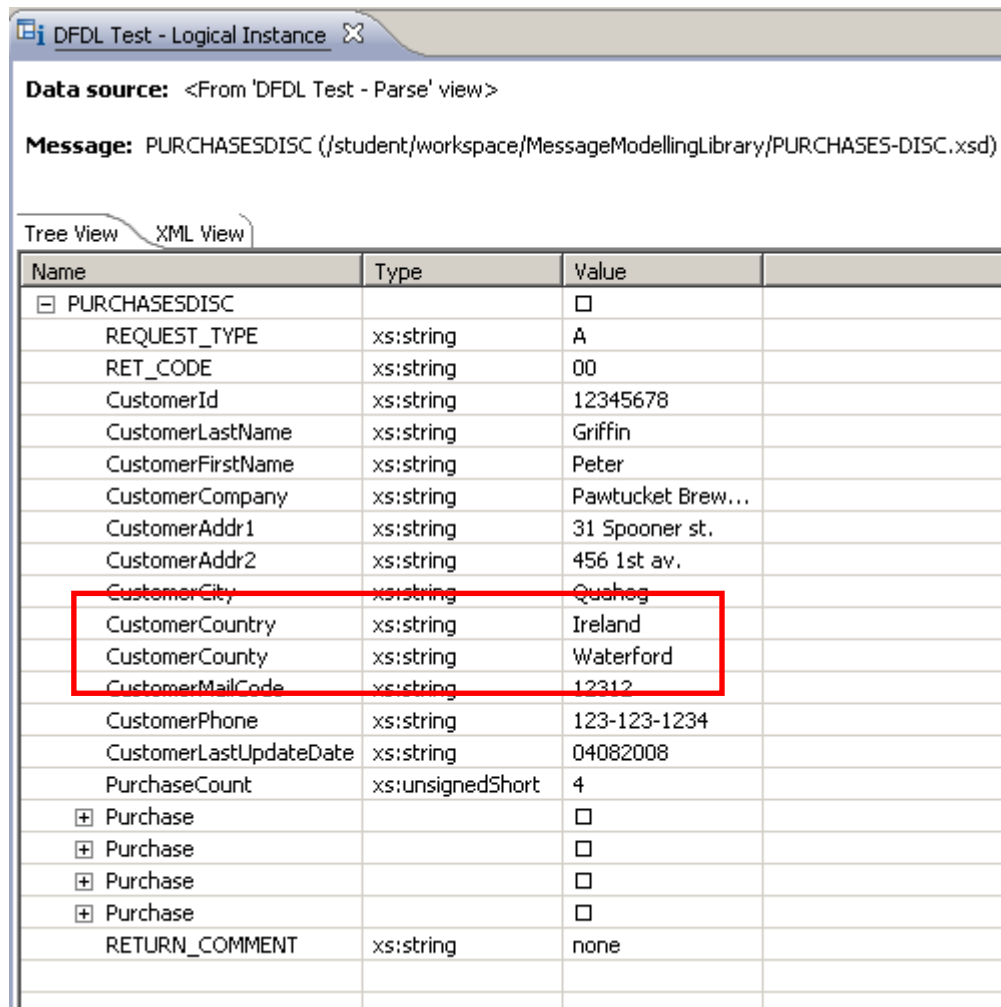
Tree View XML View

Name	Type	Value
[-] PURCHASESDISC		<input type="checkbox"/>
REQUEST_TYPE	xs:string	A
RET_CODE	xs:string	00
CustomerId	xs:string	12345678
CustomerLastName	xs:string	Griffin
CustomerFirstName	xs:string	Peter
CustomerCompany	xs:string	Pawtucket Brew...
CustomerAddr1	xs:string	31 Spooner st.
CustomerAddr2	xs:string	456 1st av.
CustomerCity	xs:string	Quehog
CustomerCountry	xs:string	UK
CustomerCounty	xs:string	Hampshire
CustomerMailCode	xs:string	12312
CustomerPhone	xs:string	123-123-1234
CustomerLastUpdateC	xs:string	04082008
PurchaseCount	xs:unsignedShort	4
[-] Purchase		<input type="checkbox"/>
[-] Purchase		<input type="checkbox"/>
[-] Purchase		<input type="checkbox"/>
[-] Purchase		<input type="checkbox"/>
RETURN_COMMENT	xs:string	none

8. Now select the input file Purchases_disc_Ireland.dat, and click on the green “Parse” button.

In the Logical Instance view, you will see that the CustomerCountry is “Ireland”.

In this case, the parser has determined that the value in this element matches one of the specified discriminators (the same one as the UK), and has used the element CustomerCounty to represent the appropriate element, and has set the value to “Waterford”, (shown highlighted below).



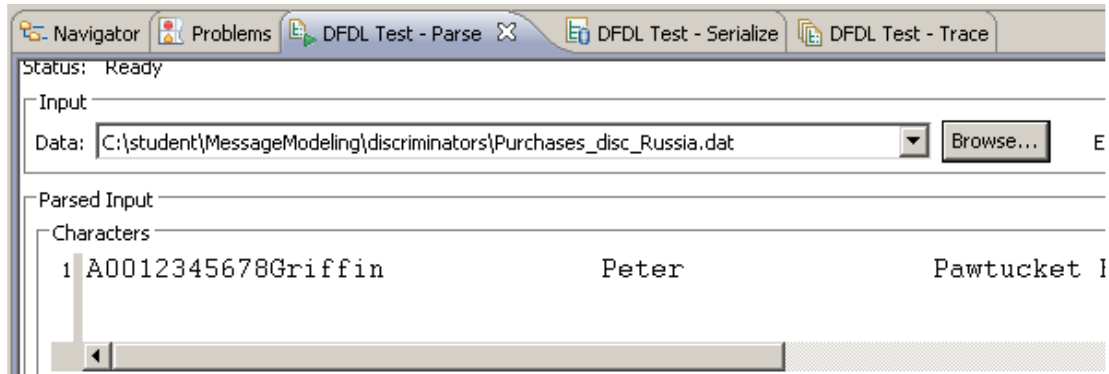
Data source: <From 'DFDL Test - Parse' view>

Message: PURCHASESDISC (/student/workspace/MessageModellingLibrary/PURCHASES-DISC.xsd)

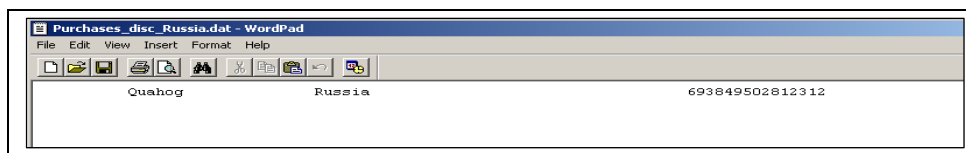
Tree View XML View

Name	Type	Value
[-] PURCHASESDISC		<input type="checkbox"/>
REQUEST_TYPE	xs:string	A
RET_CODE	xs:string	00
CustomerId	xs:string	12345678
CustomerLastName	xs:string	Griffin
CustomerFirstName	xs:string	Peter
CustomerCompany	xs:string	Pawtucket Brew...
CustomerAddr1	xs:string	31 Spooner st.
CustomerAddr2	xs:string	456 1st av.
CustomerCity	xs:string	Quahog
CustomerCountry	xs:string	Ireland
CustomerCounty	xs:string	Waterford
CustomerMailCode	xs:string	12312
CustomerPhone	xs:string	123-123-1234
CustomerLastUpdateDate	xs:string	04082008
PurchaseCount	xs:unsignedShort	4
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
RETURN_COMMENT	xs:string	none

9. Now select the input file `Purchases_disc_Russia.dat`, and click on the green “Parse” button.



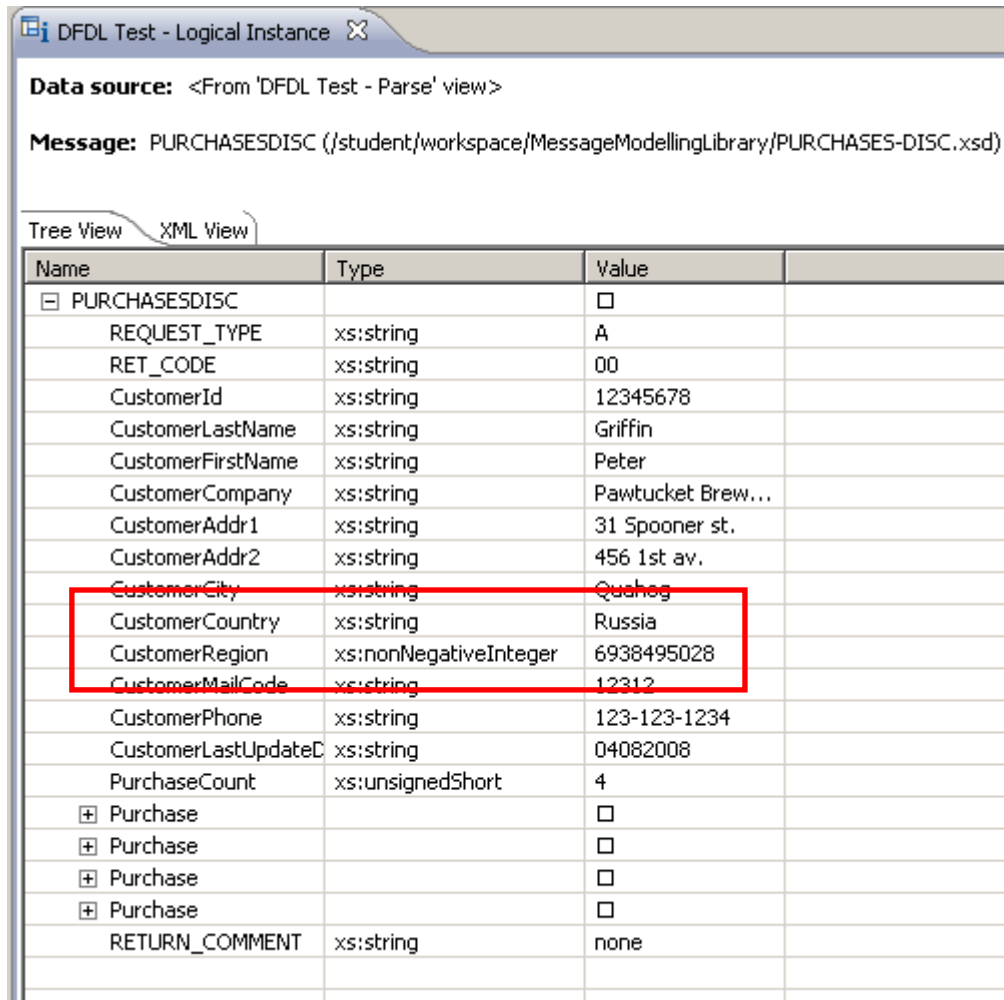
Remember that the file containing data from Russia has the data in the field “Region” as numeric, with leading blank characters.



10. In the Logical Instance view, you will see that the CustomerCountry is "Russia".

In this case, the parser has determined that the value in this element matches one of the specified discriminators (Russia), and has used the element CustomerRegion to represent the appropriate element, and has set the value to "6938495028" (shown highlighted below).

Note that the element type of CustomerRegion is nonNegativeInteger.



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

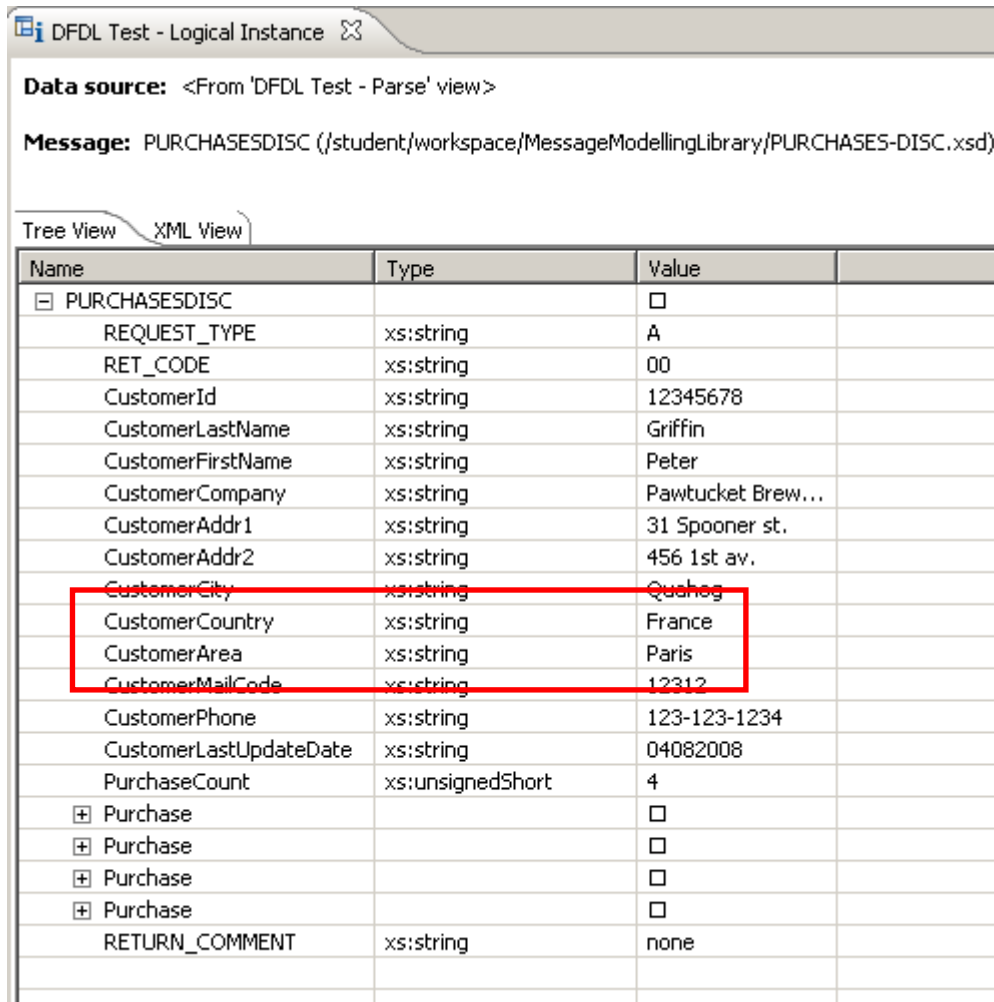
Message: PURCHASESDISC (/student/workspace/MessageModellingLibrary/PURCHASE5-DISC.xsd)

Tree View XML View

Name	Type	Value
[-] PURCHASESDISC		<input type="checkbox"/>
REQUEST_TYPE	xs:string	A
RET_CODE	xs:string	00
CustomerId	xs:string	12345678
CustomerLastName	xs:string	Griffin
CustomerFirstName	xs:string	Peter
CustomerCompany	xs:string	Pawtucket Brew...
CustomerAddr1	xs:string	31 Spooner st.
CustomerAddr2	xs:string	456 1st av.
CustomerCity	xs:string	Quahog
CustomerCountry	xs:string	Russia
CustomerRegion	xs:nonNegativeInteger	6938495028
CustomerMailCode	xs:string	12312
CustomerPhone	xs:string	123-123-1234
CustomerLastUpdateD	xs:string	04082008
PurchaseCount	xs:unsignedShort	4
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
RETURN_COMMENT	xs:string	none

11. Finally, select the input file Purchases_disc_France.dat, and click on the green “Parse” button, and run the parse again.

Note that the message model does not have any special definition for France, so the parser does not match any discriminators. The “default” choice operation is performed (ie. The last choice in the list), and the Logical Instance view shows the element CustomerArea, with the value “Paris”.



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

Message: PURCHASESDISC (/student/workspace/MessageModellingLibrary/PURCHASES-DISC.xsd)

Tree View XML View

Name	Type	Value
[-] PURCHASESDISC		<input type="checkbox"/>
REQUEST_TYPE	xs:string	A
RET_CODE	xs:string	00
CustomerId	xs:string	12345678
CustomerLastName	xs:string	Griffin
CustomerFirstName	xs:string	Peter
CustomerCompany	xs:string	Pawtucket Brew...
CustomerAddr1	xs:string	31 Spooner st.
CustomerAddr2	xs:string	456 1st av.
CustomerCity	xs:string	Quahog
CustomerCountry	xs:string	France
CustomerArea	xs:string	Paris
CustomerMailCode	xs:string	12312
CustomerPhone	xs:string	123-123-1234
CustomerLastUpdateDate	xs:string	04082008
PurchaseCount	xs:unsignedShort	4
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
[+] Purchase		<input type="checkbox"/>
RETURN_COMMENT	xs:string	none

END OF LAB GUIDE