IBM Integration Bus

Message Modeling with DFDL

Lab 6
Resolving Choices using Discriminators

June, 2013

Hands-on lab built at product
code level Version 9.0

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.

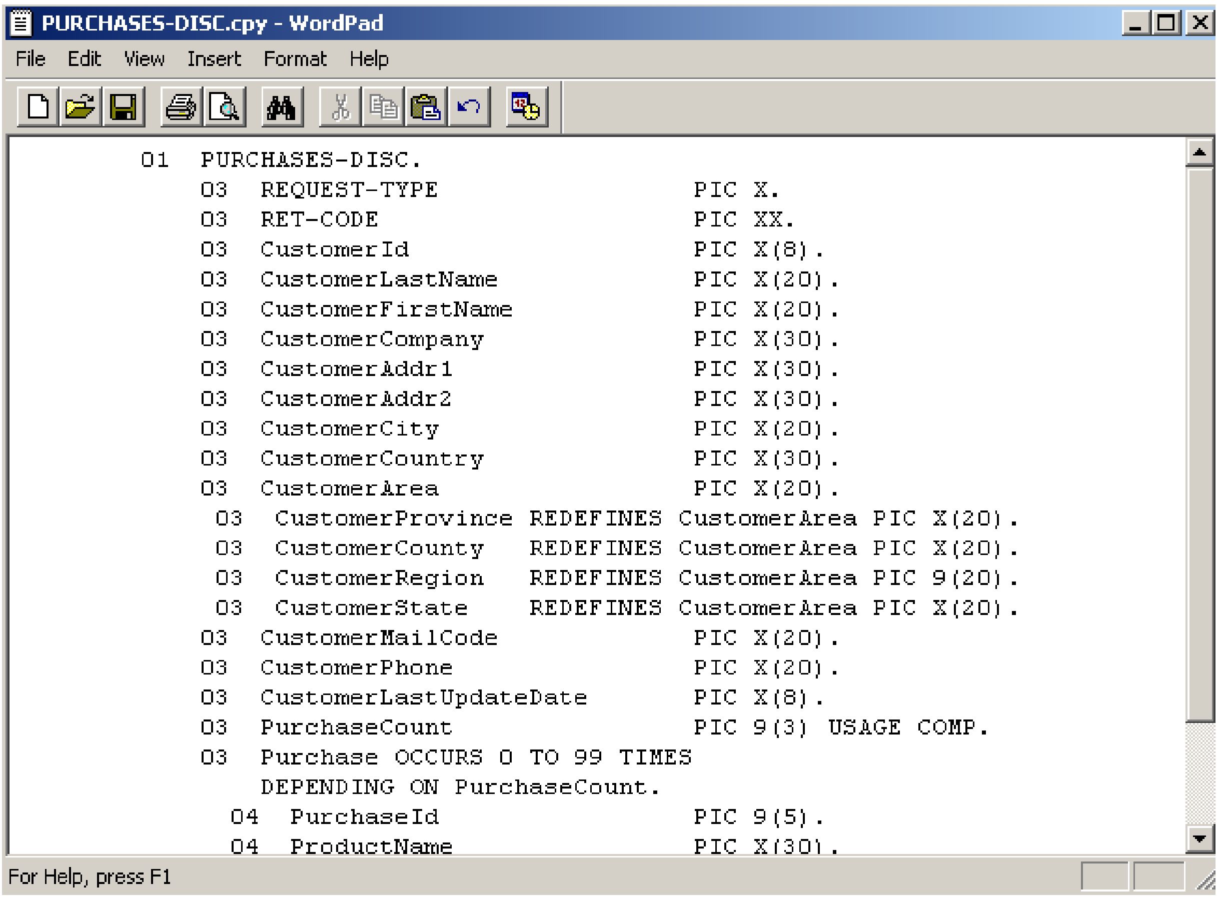
1.2 Lab Scenario

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.

- The CustomerProvince redefines clause is used when the CustomerCountry = ‘Canada’.
- The CustomerCounty redefines clause is used when the CustomerCountry = ‘UK’ or ‘Ireland’.
- 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).
- CustomerState is used when the CustomerCountry = ‘USA’
- 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.

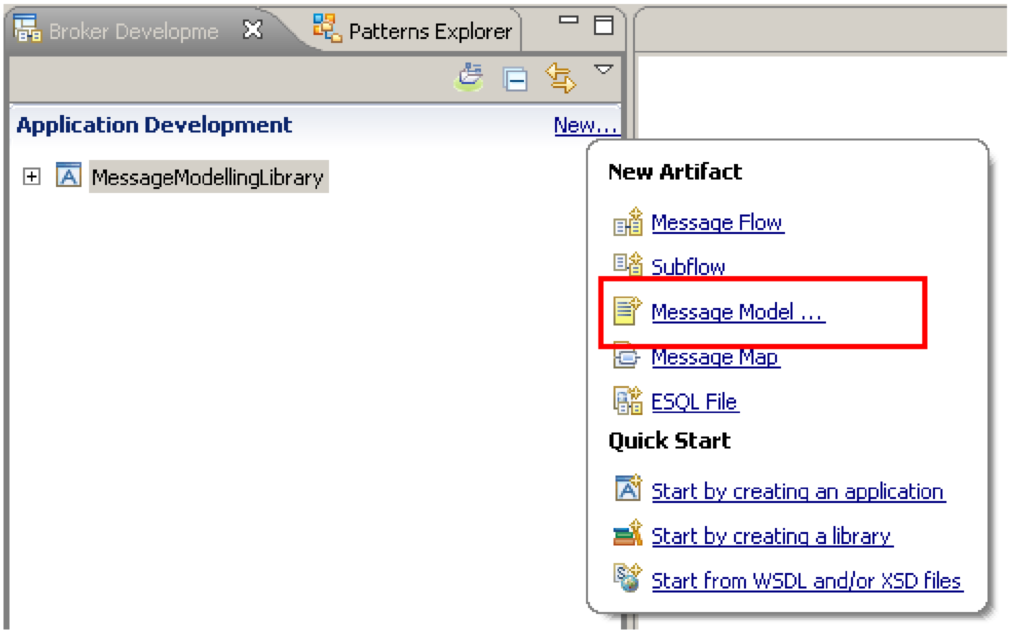

PURCHASES-DISC.cpy - WordPad
File Edit View Insert Format Help
For Help, press F1

```
        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).
```

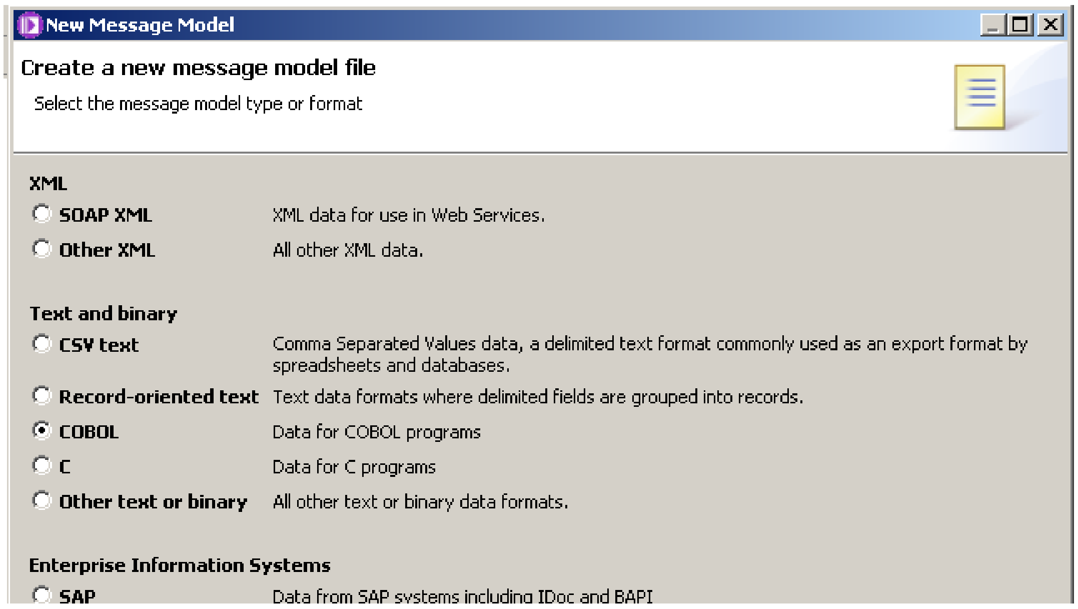
2. Build the Message Model

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.


New Message Model

Create a new message model file

Select the message model type or format

XML

○ SOAP XML	XML data for use in Web Services.
○ Other XML	All other XML data.

Text and binary

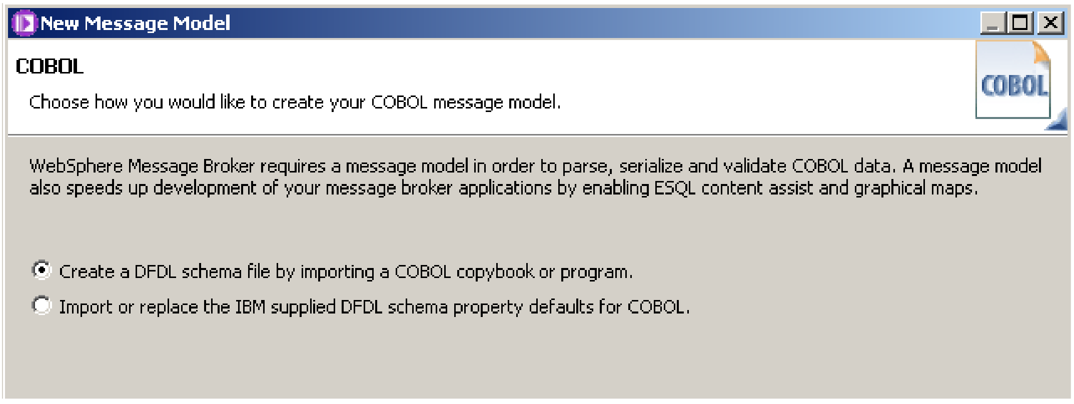
○ CSV text	Comma Separated Values data, a delimited text format commonly used as an export format by spreadsheets and databases.
○ Record-oriented text	Text data formats where delimited fields are grouped into records.
● COBOL	Data for COBOL programs
○ C	Data for C programs
○ Other text or binary	All other text or binary data formats.

Enterprise Information Systems

○ SAP	Data from SAP systems including IDoc and BAPI

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.



COBOL

Choose how you would like to create your COBOL message model.

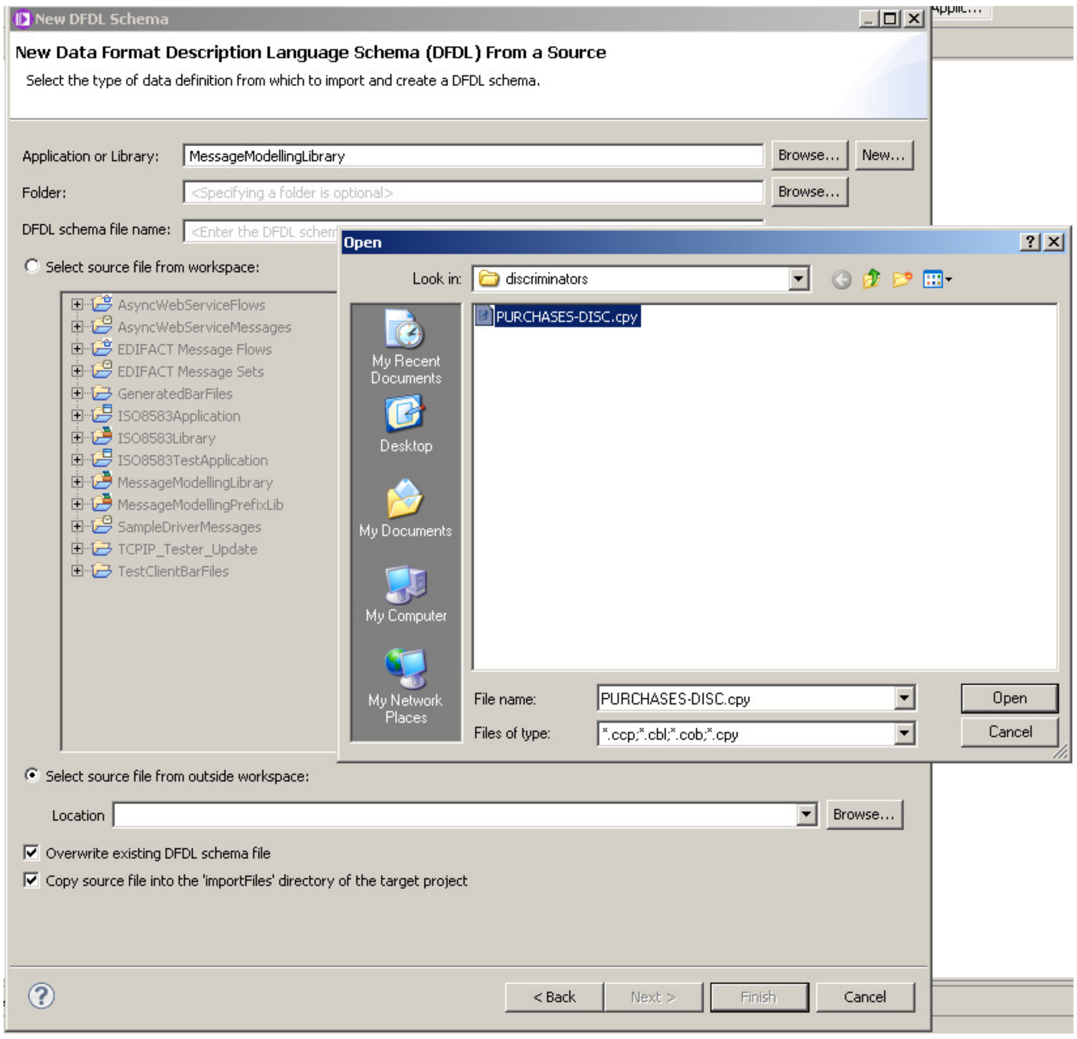
WebSphere Message Broker requires a message model in order to parse, serialize and validate COBOL data. A message model also speeds up development of your message broker applications by enabling ESQL content assist and graphical maps.

[✓ Create a DFDL schema file by importing a COBOL copybook or program.]

Import or replace the IBM supplied DFDL schema property defaults for COBOL.

4. Click “Select source from outside workspace”, and use Browse to navigate to c:\student\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: <Enter the DFDL schem

Select source file from workspace:

AsyncWebServiceFlows
AsyncWebServiceMessages
EDIFACT Message Flows
EDIFACT Message Sets
GeneratedBarFiles
ISO8583Application
ISO8583Library
ISO8583TestApplication
MessageModellingLibrary
MessageModellingPrefixLib
SampleDriverMessages
TCPIP_Tester_Update
TestClientBarFiles

Select source file from outside workspace:

Location

[✓] Overwrite existing DFDL schema file

[✓] Copy source file into the 'importFiles' directory of the target project

Open

Look in: discriminators

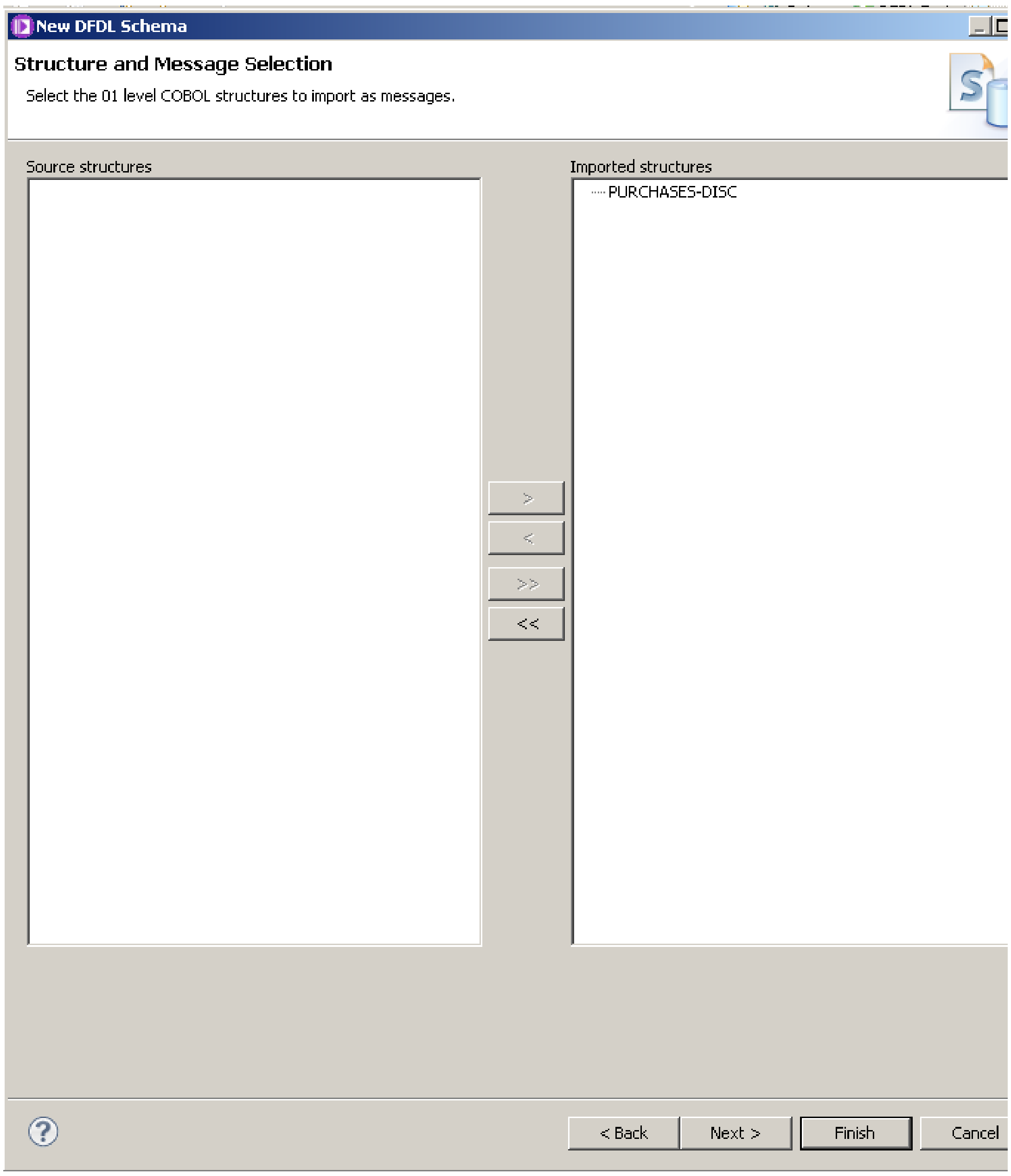
PURCHASES-DISC.cpy

File name: PURCHASES-DISC.cpy

Files of type: *.ccp;*.cbl;*.cob;*.cpy

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).



New DFDL Schema

Structure and Message Selection

Select the 01 level COBOL structures to import as messages.

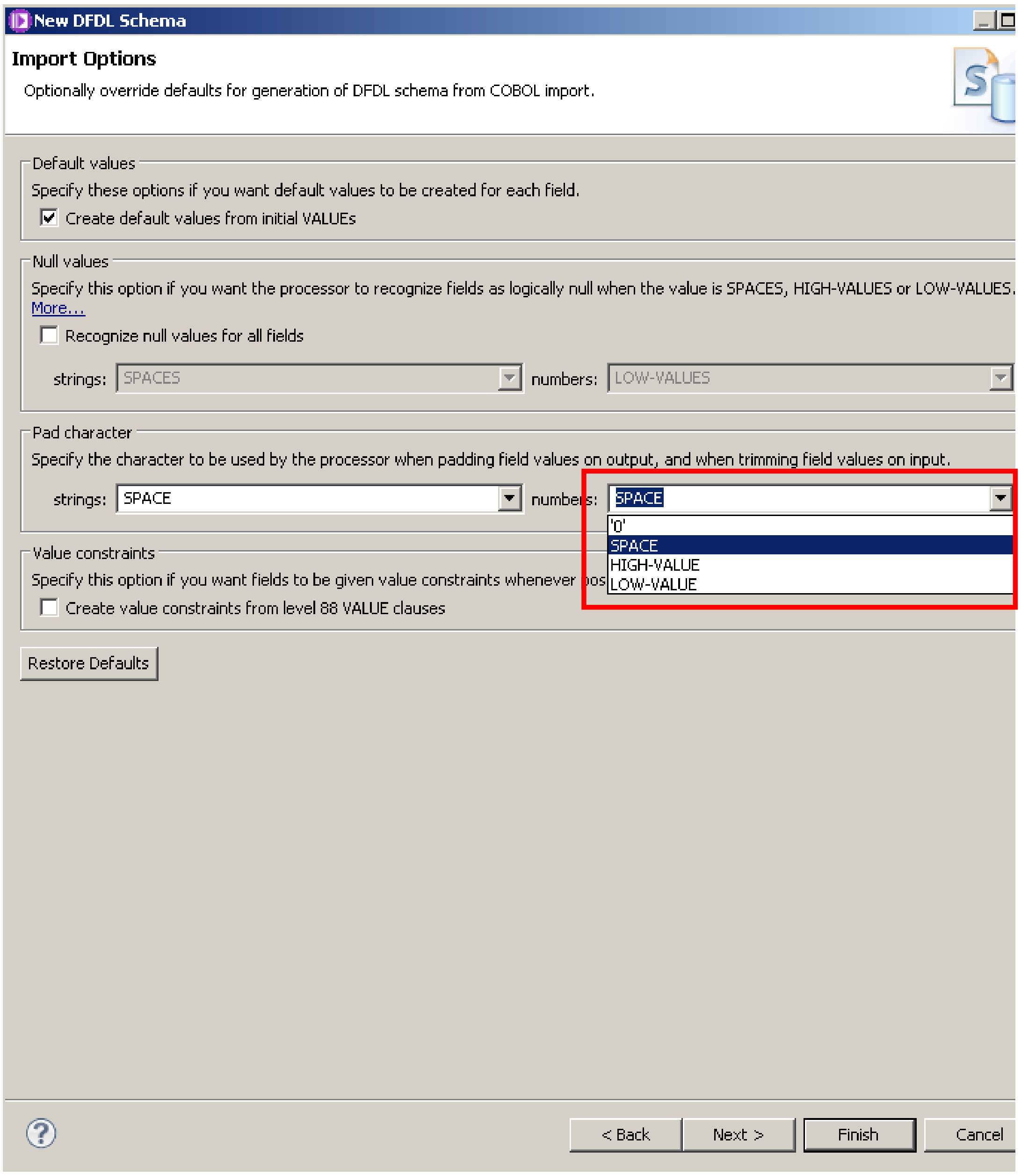
Source structures

Imported structures

PURCHASES-DISC

6. 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: SPACE

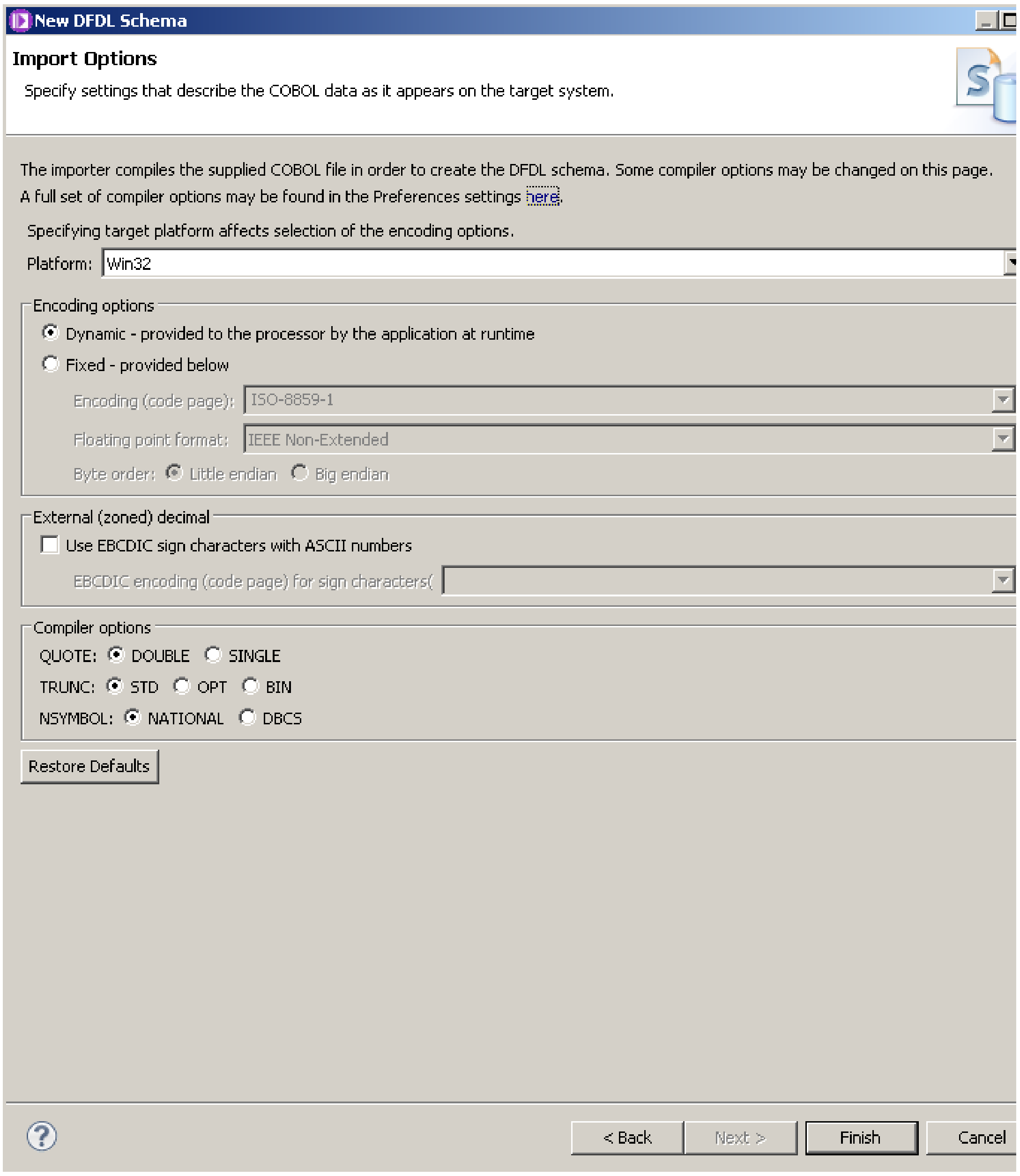
'0'
SPACE
HIGH-VALUE
LOW-VALUE

Value constraints

Specify this option if you want fields to be given value constraints whenever pos

[ ] Create value constraints from level 88 VALUE clauses

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

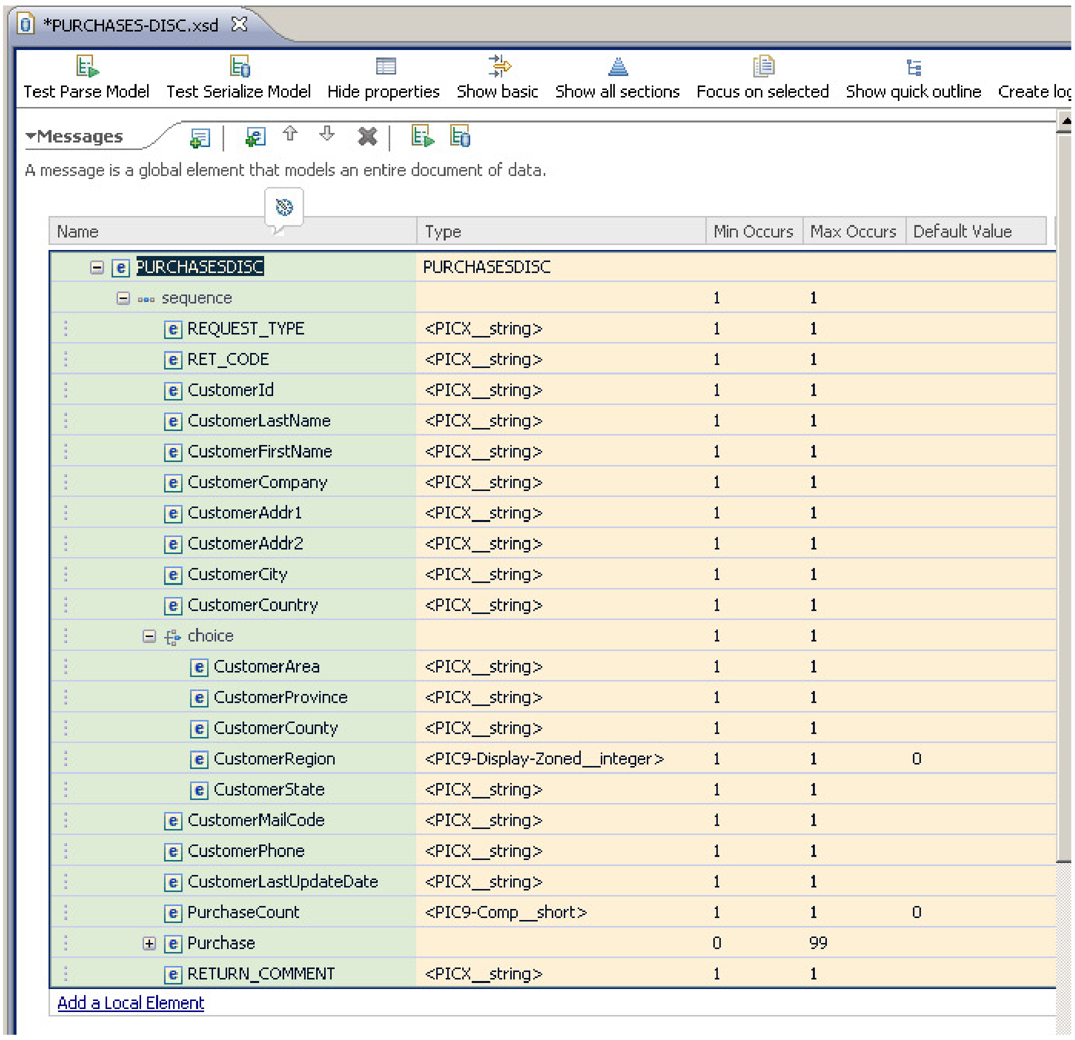
NSYMBOL: [✓ NATIONAL] DBCS

8. 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.

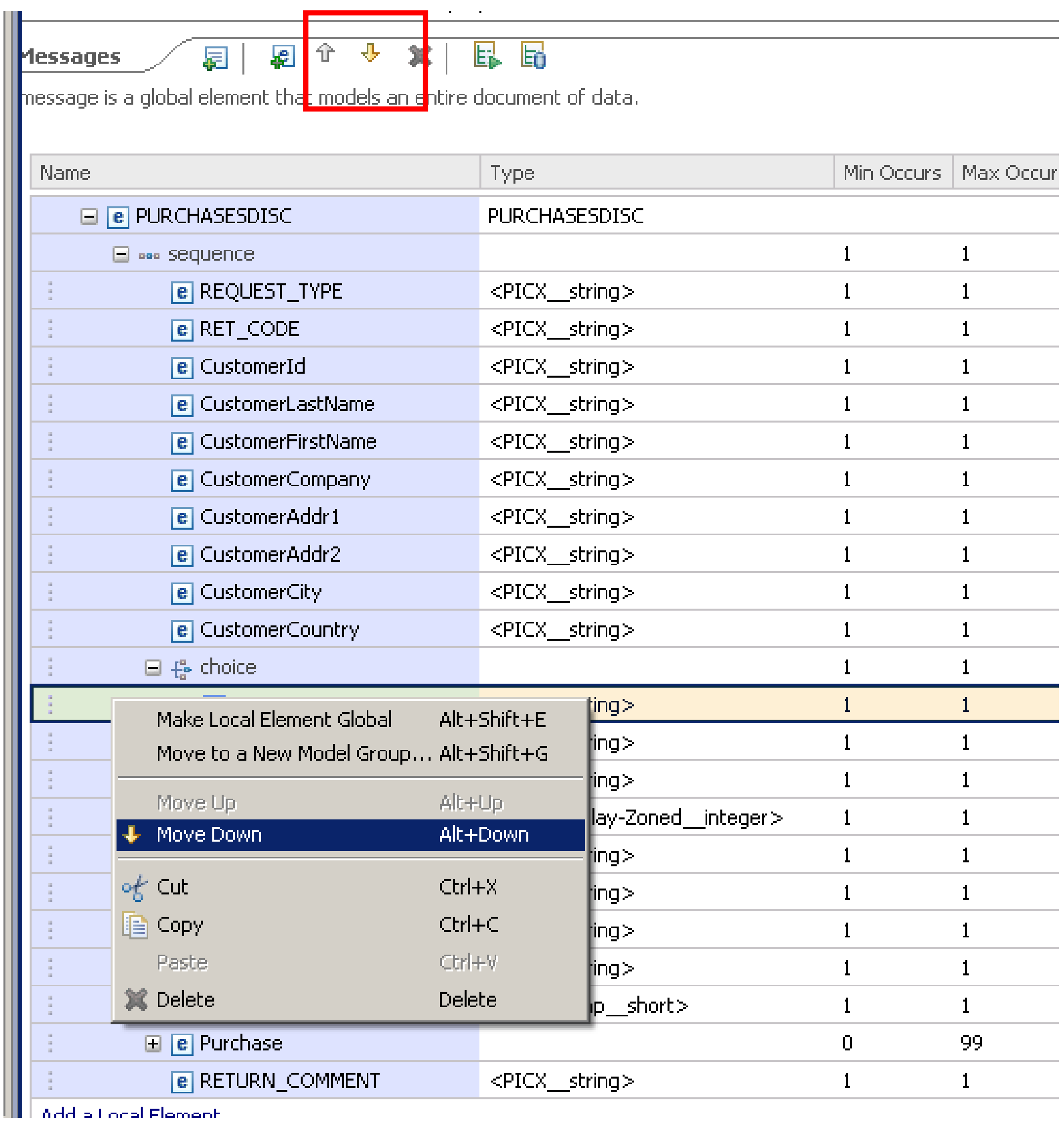


A message is a global element that models an entire document of data.

Name	Type	Min Occurs	Max Occurs	Default Value
PURCHASESDISC	PURCHASESDISC			
sequence		1	1	
REQUEST_TYPE	<PICX__string>	1	1	
RET_CODE	<PICX__string>	1	1	
CustomerId	<PICX__string>	1	1	
CustomerLastName	<PICX__string>	1	1	
CustomerFirstName	<PICX__string>	1	1	
CustomerCompany	<PICX__string>	1	1	
CustomerAddr1	<PICX__string>	1	1	
CustomerAddr2	<PICX__string>	1	1	
CustomerCity	<PICX__string>	1	1	
CustomerCountry	<PICX__string>	1	1	
choice		1	1	
CustomerArea	<PICX__string>	1	1	
CustomerProvince	<PICX__string>	1	1	
CustomerCounty	<PICX__string>	1	1	
CustomerRegion	<PIC9-Display-Zoned__integer>	1	1	0
CustomerState	<PICX__string>	1	1	
CustomerMailCode	<PICX__string>	1	1	
CustomerPhone	<PICX__string>	1	1	
CustomerLastUpdateDate	<PICX__string>	1	1	
PurchaseCount	<PIC9-Comp__short>	1	1	0
Purchase		0	99	
RETURN_COMMENT	<PICX__string>	1	1	

Add a Local Element

9. 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.



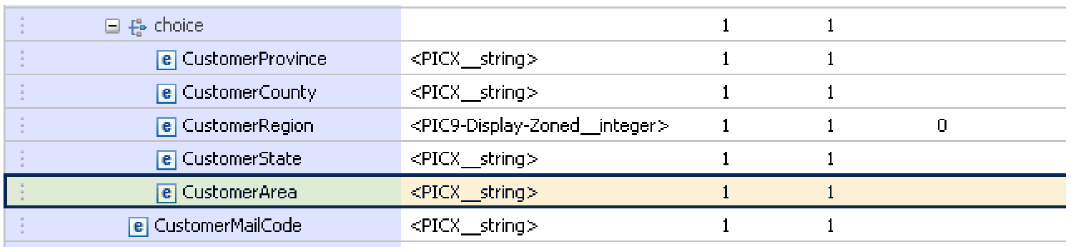
Messages

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

Name	Type	Min Occurs	Max Occur
PURCHASESDISC	PURCHASESDISC		
sequence		1	1
REQUEST_TYPE	<PICX__string>	1	1
RET_CODE	<PICX__string>	1	1
CustomerId	<PICX__string>	1	1
CustomerLastName	<PICX__string>	1	1
CustomerFirstName	<PICX__string>	1	1
CustomerCompany	<PICX__string>	1	1
CustomerAddr1	<PICX__string>	1	1
CustomerAddr2	<PICX__string>	1	1
CustomerCity	<PICX__string>	1	1
CustomerCountry	<PICX__string>	1	1
choice		1	1
	ing>	1	1
	ing>	1	1
	ing>	1	1
	lay-Zoned__integer>	1	1
	ing>	1	1
	ing>	1	1
	ing>	1	1
	ing>	1	1
	p__short>	1	1
Purchase		0	99
RETURN_COMMENT	<PICX__string>	1	1

Add a Local Element

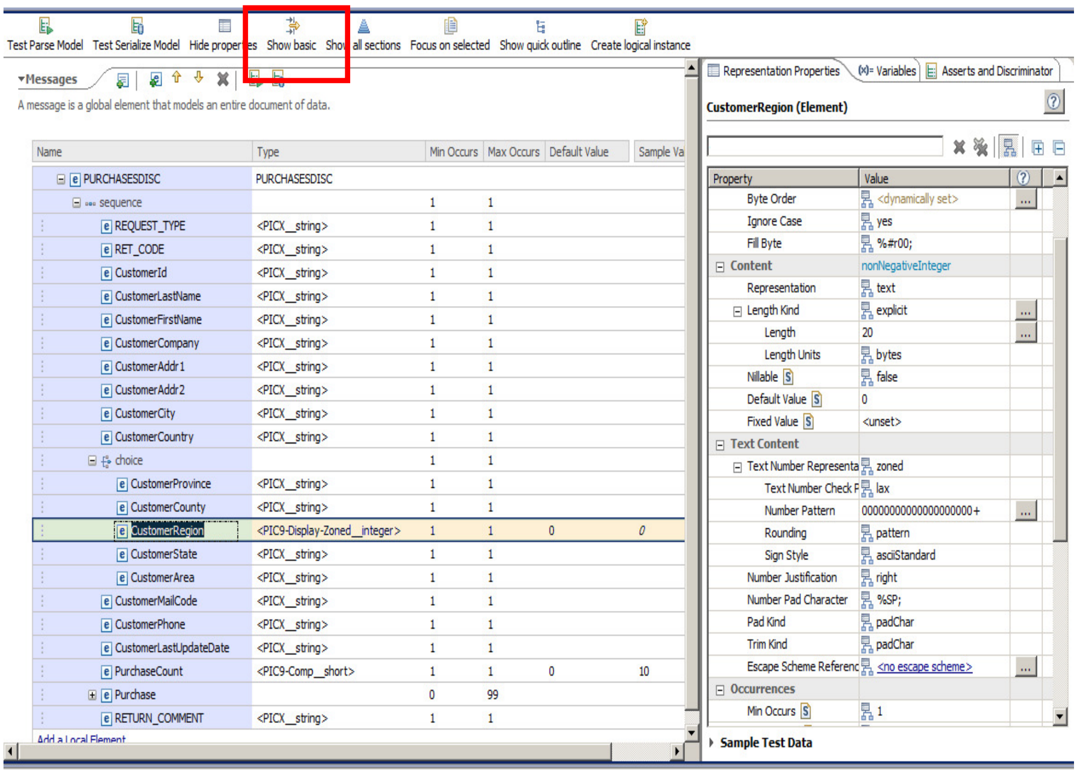
10. Move the CustomerArea element down to the bottom of the choice item. The final result should look like this:



choice		1	1	
CustomerProvince	<PICX__string>	1	1	
CustomerCounty	<PICX__string>	1	1	
CustomerRegion	<PIC9-Display-Zoned__integer>	1	1	0
CustomerState	<PICX__string>	1	1	
CustomerArea	<PICX__string>	1	1	
CustomerMailCode	<PICX__string>	1	1	

11. 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 and click on the 'Show advanced' button on the top panel. 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.



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

Name	Type	Min Occurs	Max Occurs	Default Value	Sample Va
PURCHASESDISC	PURCHASESDISC				
sequence		1	1		
REQUEST_TYPE	<PICX__string>	1	1		
RET_CODE	<PICX__string>	1	1		
CustomerId	<PICX__string>	1	1		
CustomerLastName	<PICX__string>	1	1		
CustomerFirstName	<PICX__string>	1	1		
CustomerCompany	<PICX__string>	1	1		
CustomerAddr1	<PICX__string>	1	1		
CustomerAddr2	<PICX__string>	1	1		
CustomerCity	<PICX__string>	1	1		
CustomerCountry	<PICX__string>	1	1		
choice		1	1		
CustomerProvince	<PICX__string>	1	1		
CustomerCounty	<PICX__string>	1	1		
CustomerRegion	<PIC9-Display-Zoned__integer>	1	1	0	0
CustomerState	<PICX__string>	1	1		
CustomerArea	<PICX__string>	1	1		
CustomerMailCode	<PICX__string>	1	1		
CustomerPhone	<PICX__string>	1	1		
CustomerLastUpdateDate	<PICX__string>	1	1		
PurchaseCount	<PIC9-Comp__short>	1	1	0	10
Purchase		0	99		
RETURN_COMMENT	<PICX__string>	1	1		

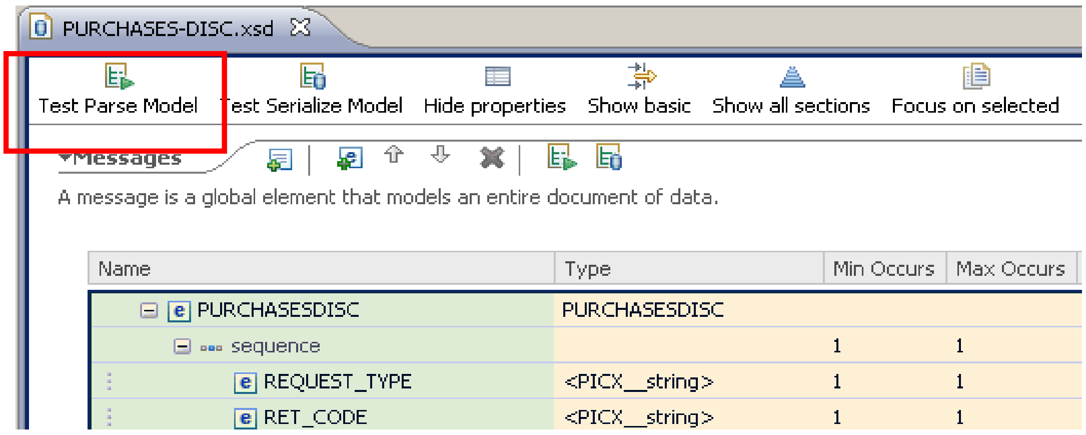
Add a Local Element

CustomerRegion (Element)

Property	Value
Byte Order	<dynamically set>
Ignore Case	yes
Fill Byte	%#r00;
Content	nonNegativeInteger
Representation	text
Length Kind	explicit
Length	20
Length Units	bytes
Nillable	false
Default Value	0
Fixed Value	<unset>
Text Content	
Text Number Representa	zoned
Text Number Check P	lax
Number Pattern	0000000000000000000+
Rounding	pattern
Sign Style	asciiStandard
Number Justification	right
Number Pad Character	%SP;
Pad Kind	padChar
Trim Kind	padChar
Escape Scheme Referenc	<no escape scheme>
Occurrences	
Min Occurs	1

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

    Click Test Parse Model.



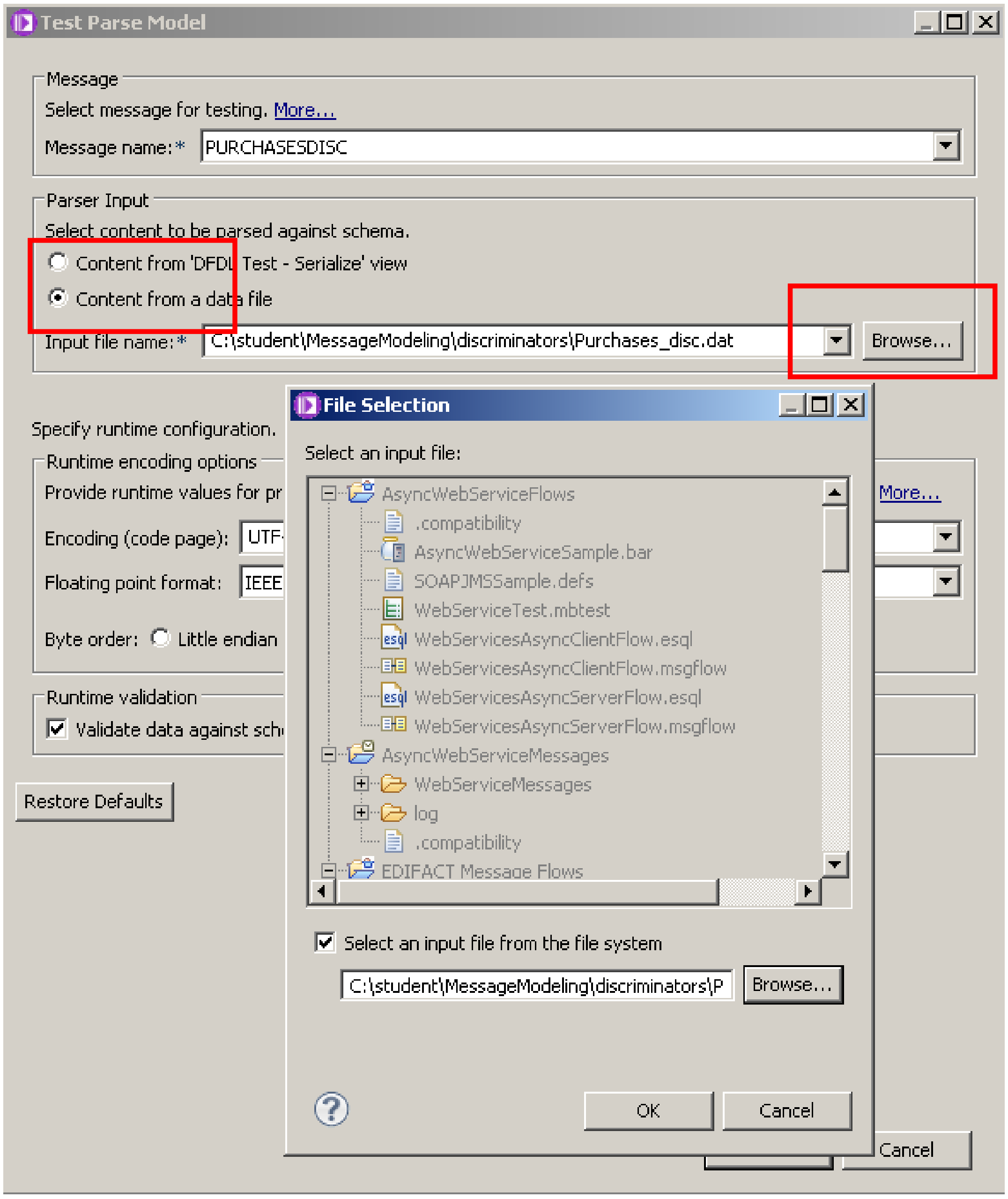
A message is a global element that models an entire document of data.

Name	Type	Min Occurs	Max Occurs
PURCHASESDISC	PURCHASESDISC		
sequence		1	1
REQUEST_TYPE	<PICX__string>	1	1
RET_CODE	<PICX__string>	1	1

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.



Test Parse Model

Message

Select message for testing. More...

Message name:* PURCHASESDISC

Parser Input

Select content to be parsed against schema.

( ) Content from 'DFDL Test - Serialize' view

(●) Content from a data file

Input file name:* C:\student\MessageModeling\discriminators\Purchases_disc.dat

Specify runtime configuration.

Runtime encoding options

Provide runtime values for pr

Encoding (code page): UTF-

Floating point format: IEEE

Byte order: ( ) Little endian

Runtime validation

[✓] Validate data against sch

File Selection

Select an input file:

- AsyncWebServiceFlows
  - .compatibility
  - AsyncWebServiceSample.bar
  - SOAPJMSSample.defs
  - WebServiceTest.mbtest
  - WebServicesAsyncClientFlow.esql
  - WebServicesAsyncClientFlow.msgflow
  - WebServicesAsyncServerFlow.esql
  - WebServicesAsyncServerFlow.msgflow
- AsyncWebServiceMessages
  - WebServiceMessages
  - log
  - .compatibility
- EDIFACT Message Flows

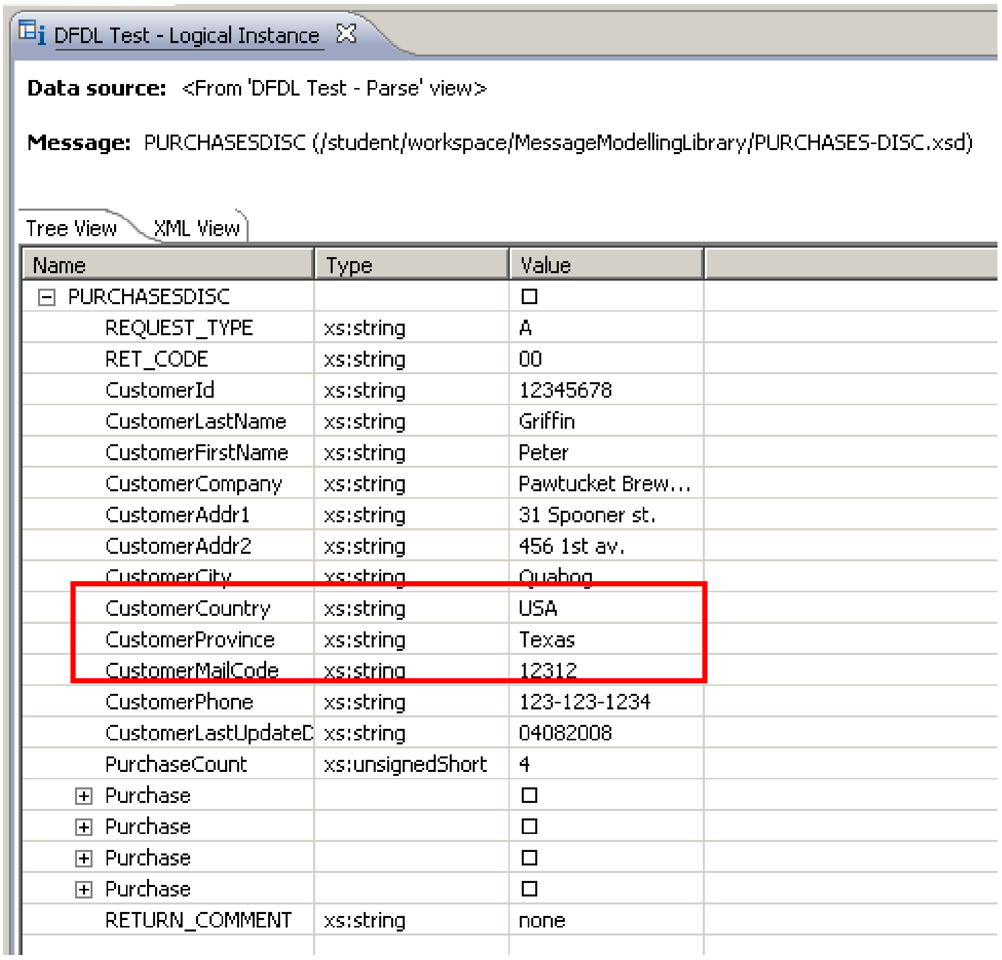
[✓] Select an input file from the file system

C:\student\MessageModeling\discriminators\P

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)

Name	Type	Value
⊟ PURCHASESDISC		☐
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
CustomerProvince	xs:string	Texas
CustomerMailCode	xs:string	12312
CustomerPhone	xs:string	123-123-1234
CustomerLastUpdateD	xs:string	04082008
PurchaseCount	xs:unsignedShort	4
⊞ Purchase		☐
⊞ Purchase		☐
⊞ Purchase		☐
⊞ Purchase		☐
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.

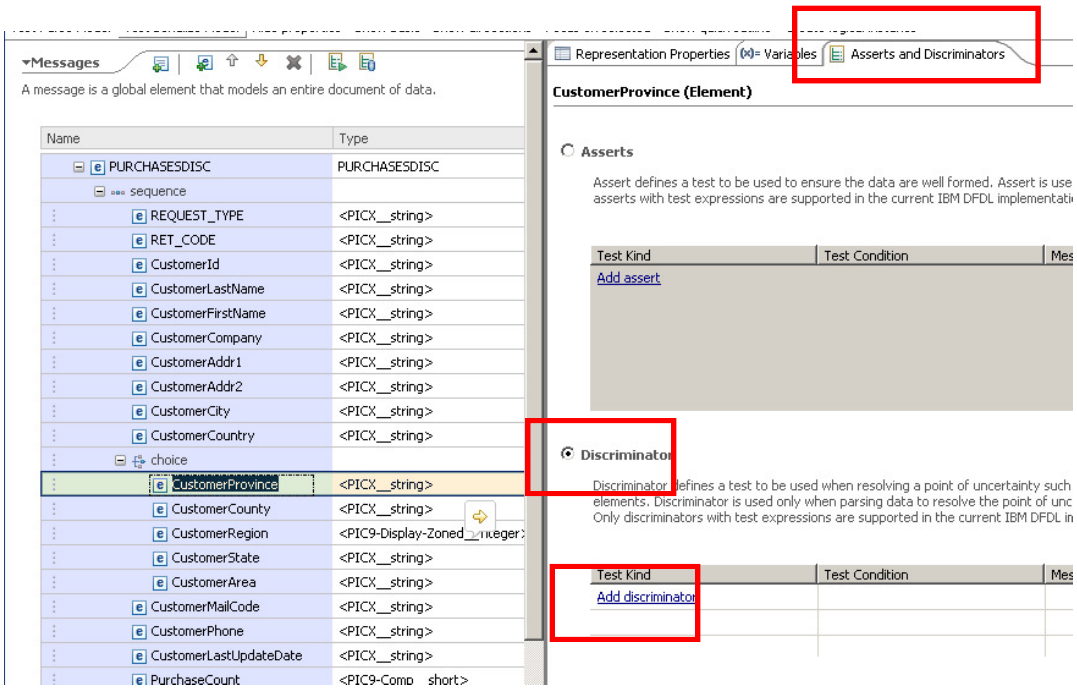
This will be done by setting a discriminator on the CustomerProvince, CustomerCounty, CustomerRegion and CustomerState elements. No change will be made to the CustomerArea element, which will act as a default for all countries that do not have explicit discriminators.

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.



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

Name	Type
PURCHASESDISC	PURCHASESDISC
sequence	
REQUEST_TYPE	<PICX__string>
RET_CODE	<PICX__string>
CustomerId	<PICX__string>
CustomerLastName	<PICX__string>
CustomerFirstName	<PICX__string>
CustomerCompany	<PICX__string>
CustomerAddr1	<PICX__string>
CustomerAddr2	<PICX__string>
CustomerCity	<PICX__string>
CustomerCountry	<PICX__string>
choice	
CustomerProvince	<PICX__string>
CustomerCounty	<PICX__string>
CustomerRegion	<PIC9-Display-Zoned_integer>
CustomerState	<PICX__string>
CustomerArea	<PICX__string>
CustomerMailCode	<PICX__string>
CustomerPhone	<PICX__string>
CustomerLastUpdateDate	<PICX__string>
PurchaseCount	<PIC9-Comp short>

CustomerProvince (Element)

○ Asserts

Assert defines a test to be used to ensure the data are well formed. Assert is use
asserts with test expressions are supported in the current IBM DFDL implementati

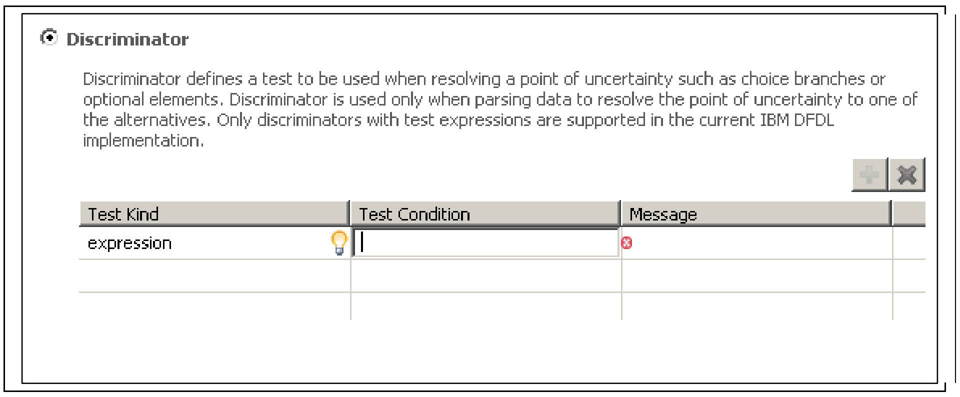
Test Kind	Test Condition	Mes
Add assert		

● Discriminator

Discriminator defines a test to be used when resolving a point of uncertainty such
elements. Discriminator is used only when parsing data to resolve the point of unc
Only discriminators with test expressions are supported in the current IBM DFDL in

Test Kind	Test Condition	Mes
Add discriminator		

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

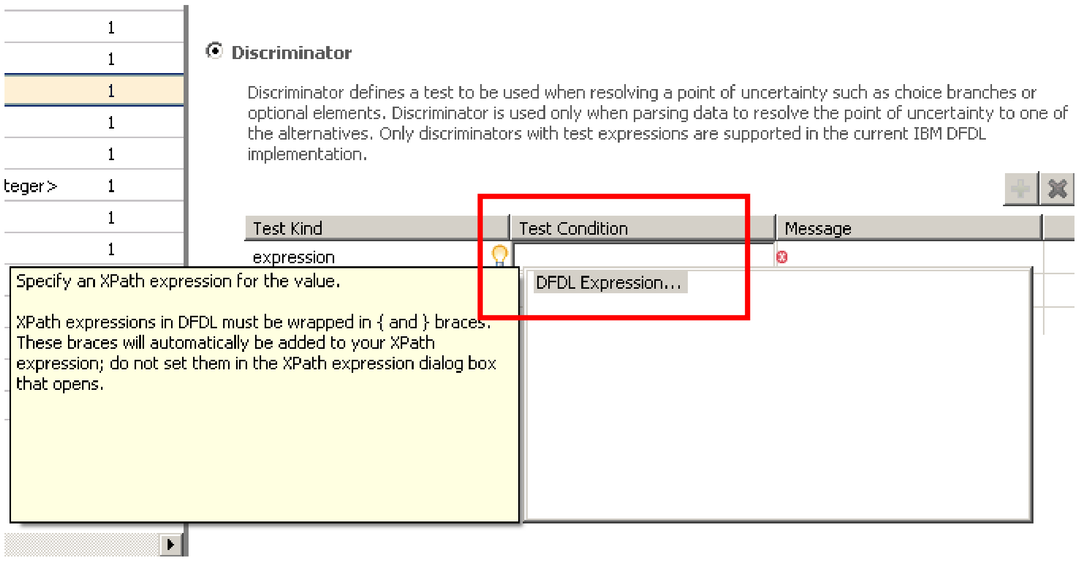


Discriminator

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 DFDL implementation.

Test Kind	Test Condition	Message
expression		

3. 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”.



Discriminator

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 DFDL implementation.

Test Kind	Test Condition	Message
expression		

Specify an XPath expression for the value.

XPath expressions in DFDL must be wrapped in { and } braces. These braces will automatically be added to your XPath expression; do not set them in the XPath expression dialog box that opens.

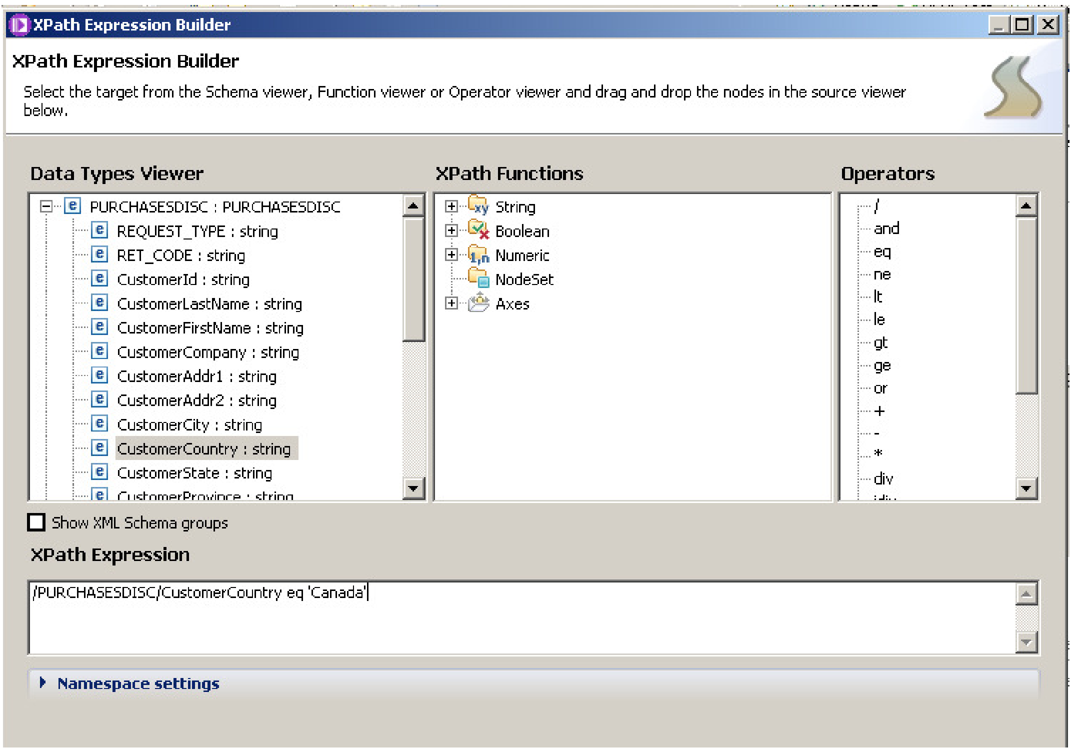
4. This will open the XPath Expression Builder.

   We will generate an XPath expression that will check the CustomerCountry element for the “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.

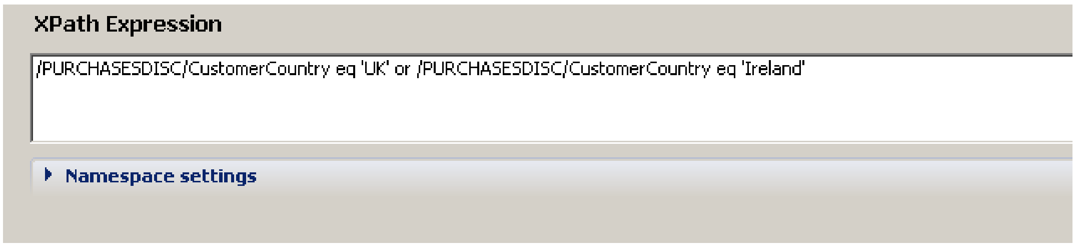


/PURCHASESDISC/CustomerCountry eq 'Canada'

5. Performs steps 1 – 4 for the element CustomerCounty. In this case, CustomerCounty 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 CustomerCounty should be:

```
/PURCHASESDISC/CustomerCountry eq ‘UK’ or
/PURCHASESDISC/CustomerCountry eq ‘Ireland’
```



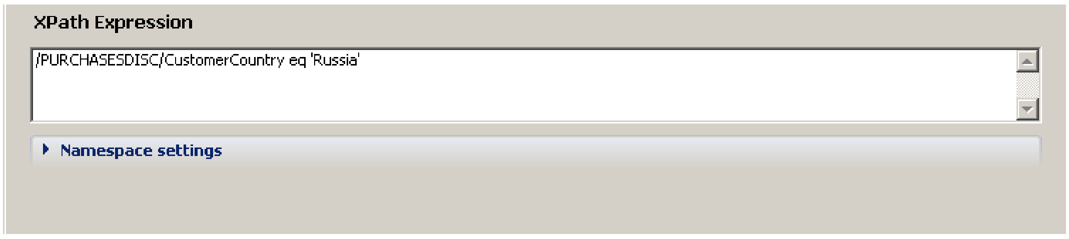
/PURCHASESDISC/CustomerCountry eq 'UK' or /PURCHASESDISC/CustomerCountry eq 'Ireland'

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.

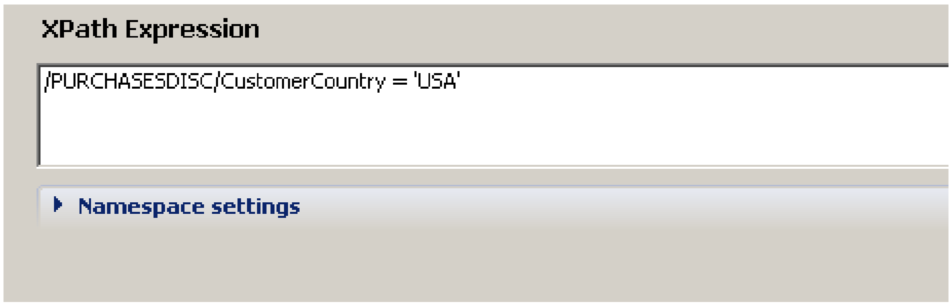


```
/PURCHASESDISC/CustomerCountry eq 'Russia'
```

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’
   ```



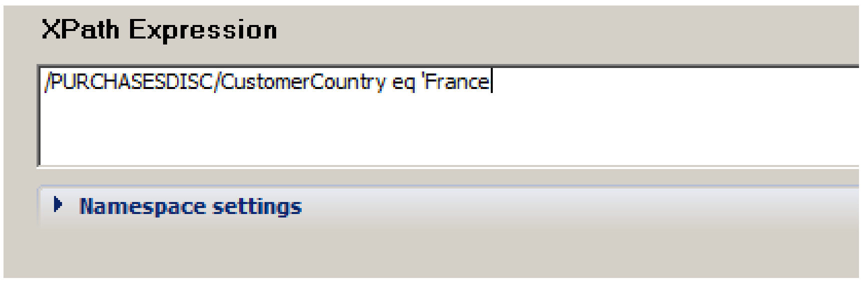
```
/PURCHASESDISC/CustomerCountry = 'USA'
```

8. Performs steps 1 – 4 for the element CustomerArea. In this case, CustomerArea should be

used to parse the element if the CustomerArea is “France”.

The XPath expression for the element CustomerArea should be:

```
/PURCHASESDISC/CustomerCountry eq ‘France’
```

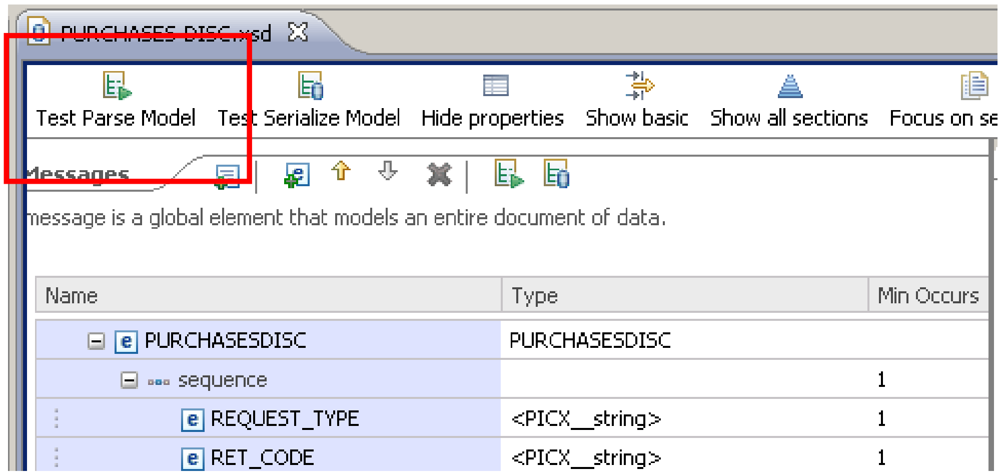


/PURCHASESDISC/CustomerCountry eq 'France

9. Save the model

2.3 Test the model with discriminators

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



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

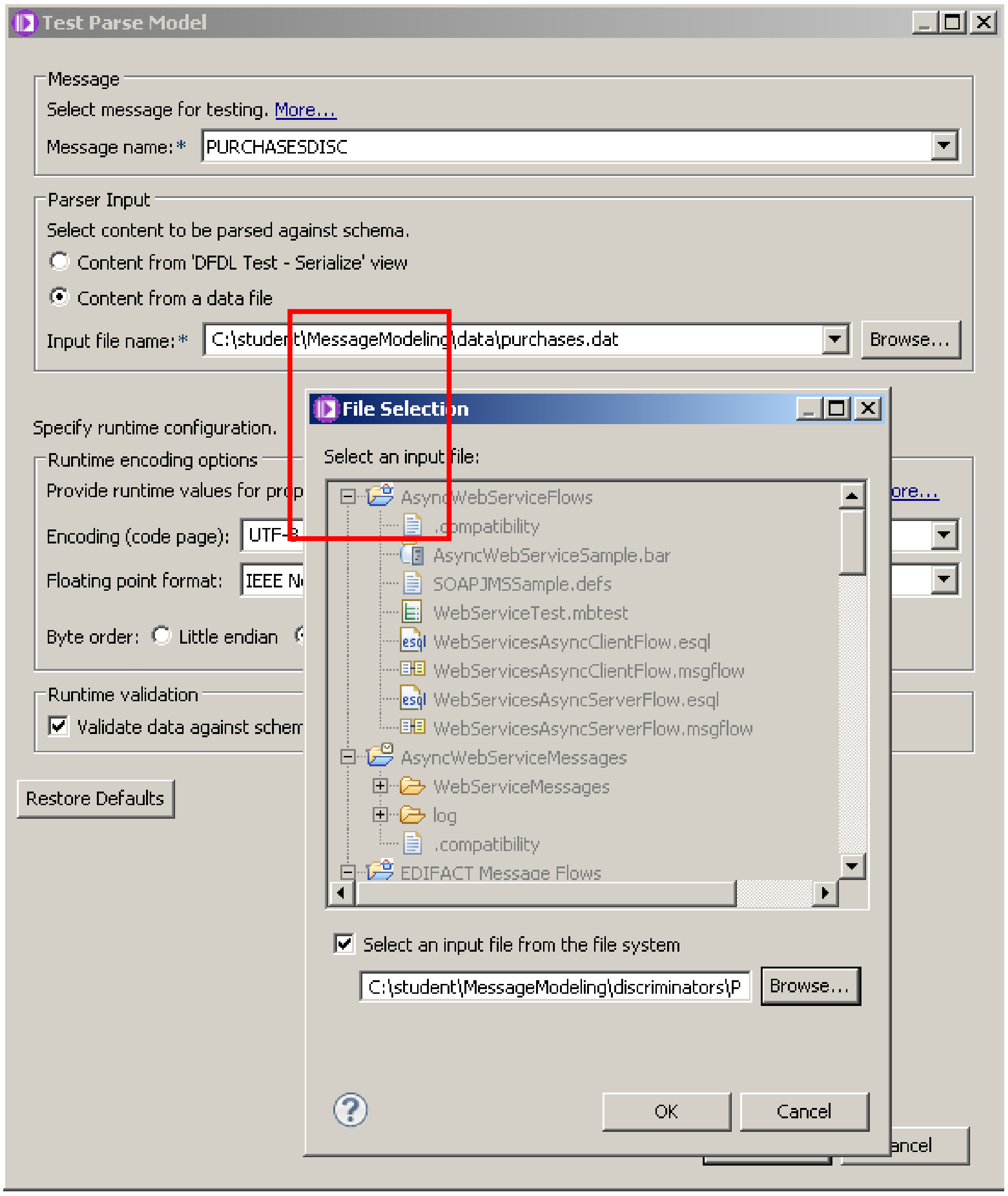
Name	Type	Min Occurs
PURCHASESDISC	PURCHASESDISC	
sequence		1
REQUEST_TYPE	<PICX__string>	1
RET_CODE	<PICX__string>	1

2. Select “Content from a data file”, click Browse, and select an input file from the file system.

   Navigate to c:\student\messagemodelling\discriminators.

   Select the file Purchases_disc_USA.dat.

   Click OK and OK.

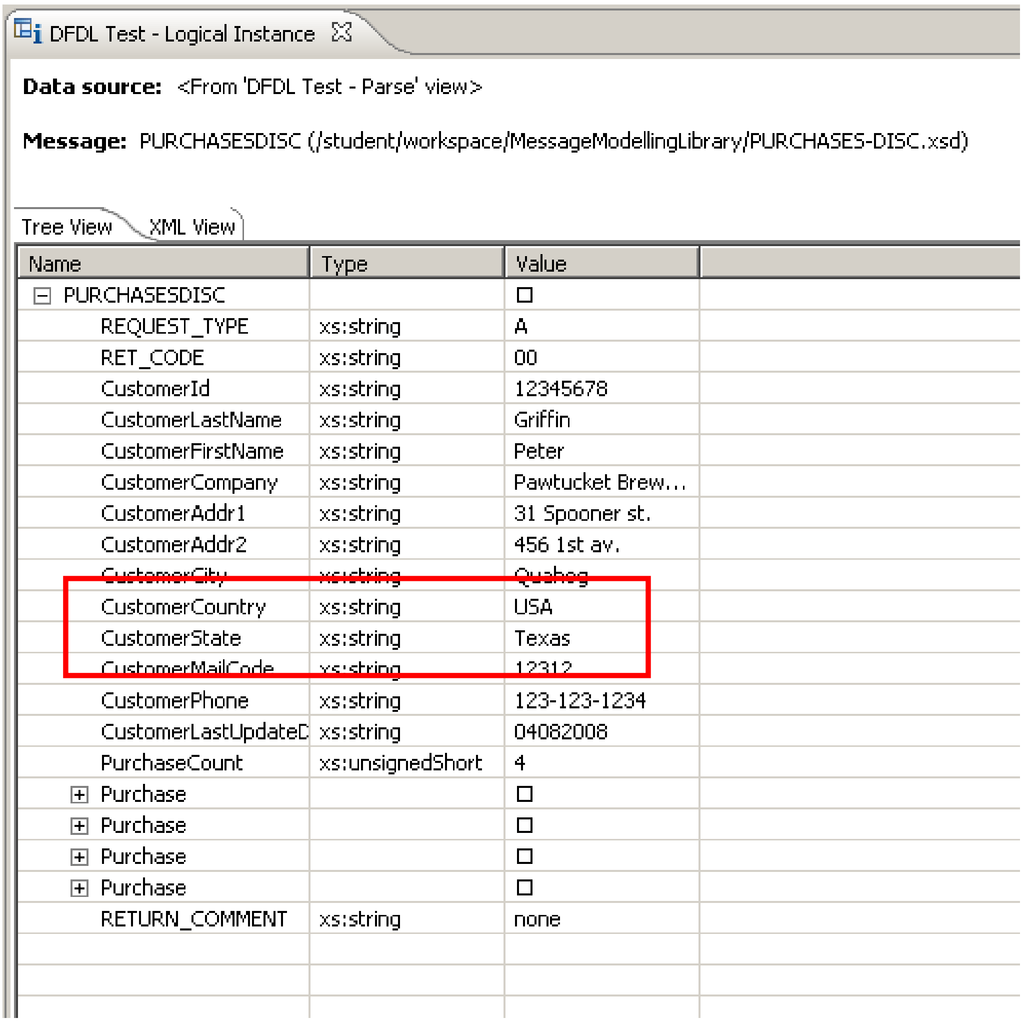

Test Parse Model
Message
Select message for testing. More...
Message name:* PURCHASESDISC
Parser Input
Select content to be parsed against schema.
Content from 'DFDL Test - Serialize' view
Content from a data file
Input file name:* C:\student\MessageModeling\data\purchases.dat
Browse...
Specify runtime configuration.
Runtime encoding options
Provide runtime values for pro
Encoding (code page): UTF-8
Floating point format: IEEE N
Byte order: Little endian
Runtime validation
Validate data against schem
Restore Defaults
File Selection
Select an input file:
AsyncWebServiceFlows
.compatibility
AsyncWebServiceSample.bar
SOAPJMSSample.defs
WebServiceTest.mbtest
WebServicesAsyncClientFlow.esql
WebServicesAsyncClientFlow.msgflow
WebServicesAsyncServerFlow.esql
WebServicesAsyncServerFlow.msgflow
AsyncWebServiceMessages
WebServiceMessages
log
.compatibility
EDIFACT Message Flows
Select an input file from the file system
C:\student\MessageModeling\discriminators\P
Browse...
OK
Cancel

Click OK then OK.

3. 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 CustomerArea..



DFDL Test - Logical Instance

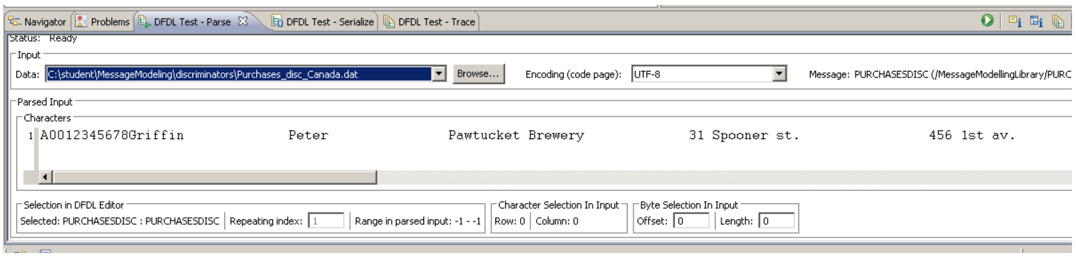
Data source: <From 'DFDL Test - Parse' view>

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

Tree View | XML View

Name	Type	Value
⊟ PURCHASESDISC		☐
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
CustomerLastUpdateD	xs:string	04082008
PurchaseCount	xs:unsignedShort	4
⊞ Purchase		☐
⊞ Purchase		☐
⊞ Purchase		☐
⊞ Purchase		☐
RETURN_COMMENT	xs:string	none

4. Now select the input file Purchases_disc_Canada.dat, and click on the green "Parse" button.



Navigator | Problems | DFDL Test - Parse | DFDL Test - Serialize | DFDL Test - Trace

Status: Ready

Input

Data: C:\student\MessageModeling\discriminators\Purchases_disc_Canada.dat   Browse...   Encoding (code page): UTF-8   Message: PURCHASESDISC (/MessageModellingLibrary/PURC

Parsed Input

Characters

```
1 A00123456789Griffin            Peter               Pawtucket Brewery          31 Spooner st.            456 1st av.
```

Selection in DFDL Editor: Selected: PURCHASESDISC : PURCHASESDISC | Repeating index: 1 | Range in parsed input: -1 - -1

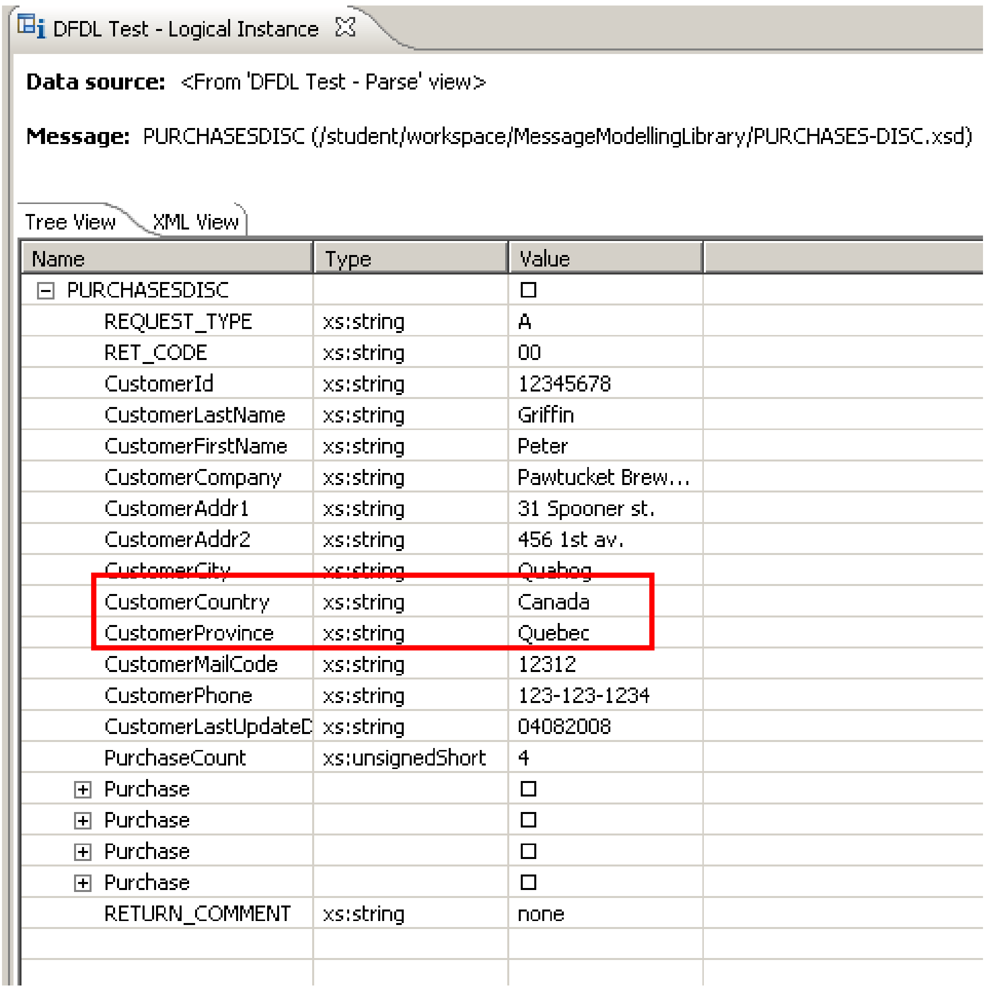
Character Selection In Input: Row: 0 | Column: 0

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

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".



DFDL Test - Logical Instance

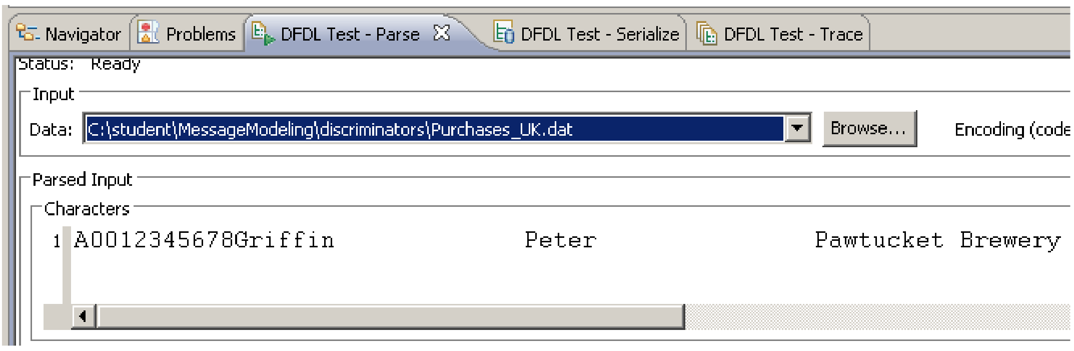
Data source: <From 'DFDL Test - Parse' view>

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

Tree View | XML View

Name	Type	Value
⊟ PURCHASESDISC		☐
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	Canada
CustomerProvince	xs:string	Quebec
CustomerMailCode	xs:string	12312
CustomerPhone	xs:string	123-123-1234
CustomerLastUpdateD	xs:string	04082008
PurchaseCount	xs:unsignedShort	4
⊞ Purchase		☐
⊞ Purchase		☐
⊞ Purchase		☐
⊞ Purchase		☐
RETURN_COMMENT	xs:string	none

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

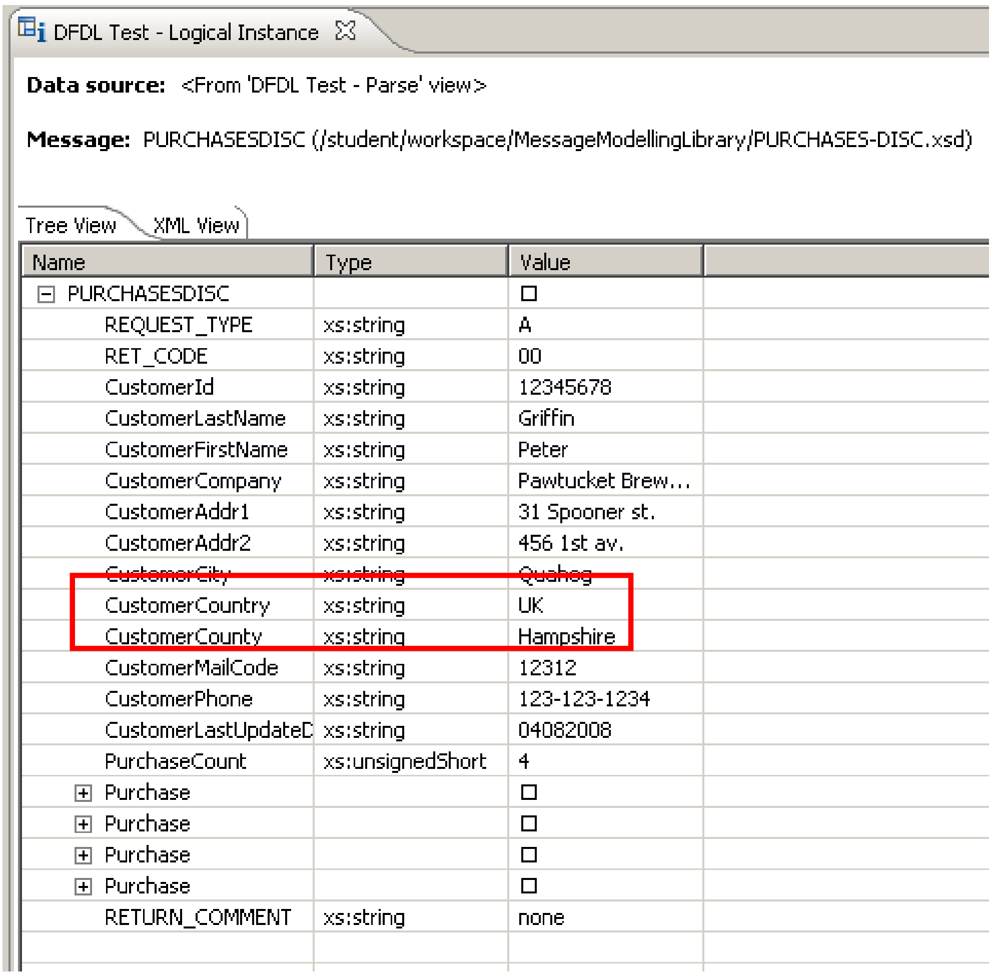


```
C:\student\MessageModeling\discriminators\Purchases_UK.dat
```

```
1 A0012345678Griffin             Peter               Pawtucket Brewery
```

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 CustomerCounty 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).



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

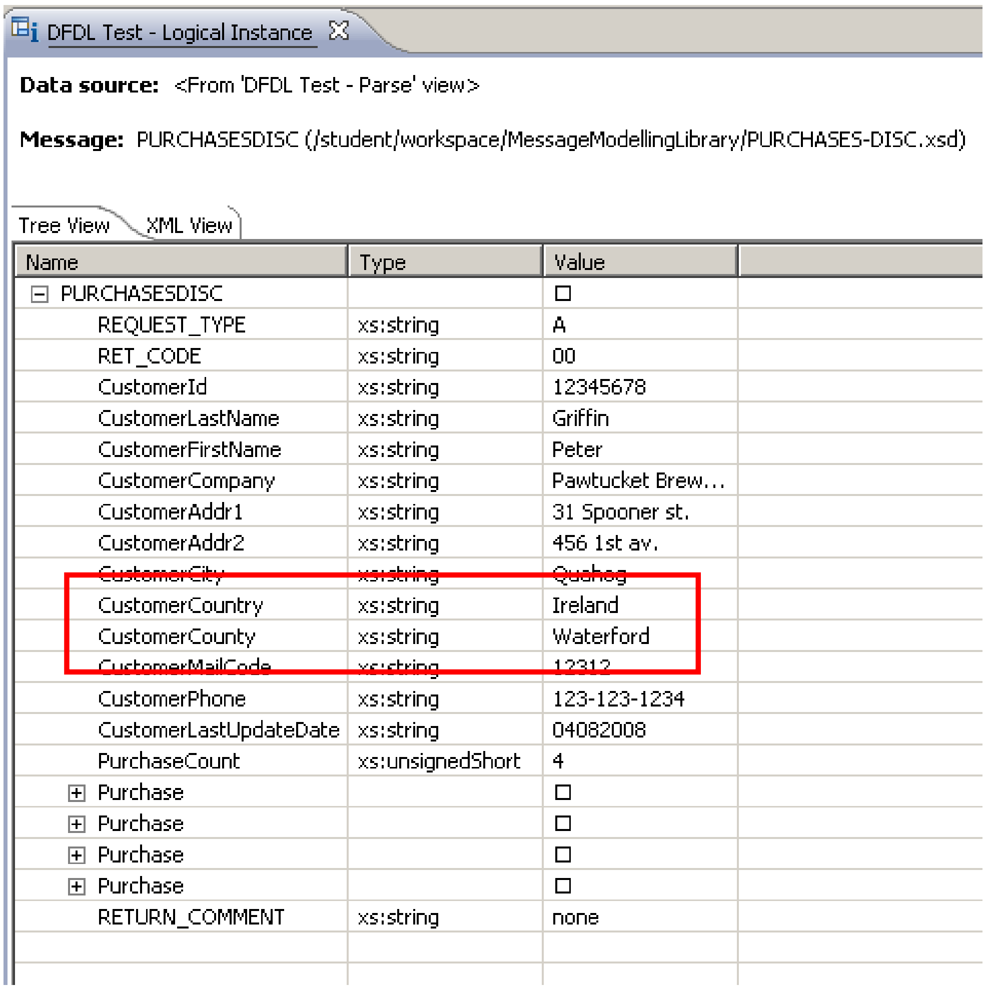
Message: PURCHASESDISC (/student/workspace/MessageModellingLibrary/PURCHASES-DISC.xsd)

Name	Type	Value
⊟ PURCHASESDISC		☐
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	UK
CustomerCounty	xs:string	Hampshire
CustomerMailCode	xs:string	12312
CustomerPhone	xs:string	123-123-1234
CustomerLastUpdateD	xs:string	04082008
PurchaseCount	xs:unsignedShort	4
⊞ Purchase		☐
⊞ Purchase		☐
⊞ Purchase		☐
⊞ Purchase		☐
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).

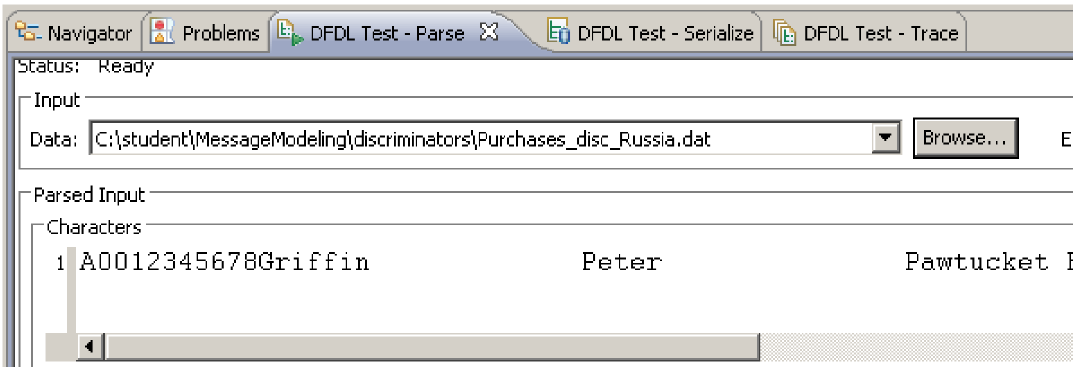


Data source: <From 'DFDL Test - Parse' view>

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

Name	Type	Value
⊟ PURCHASESDISC		☐
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		☐
⊞ Purchase		☐
⊞ Purchase		☐
⊞ Purchase		☐
RETURN_COMMENT	xs:string	none

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



Status: Ready

Input

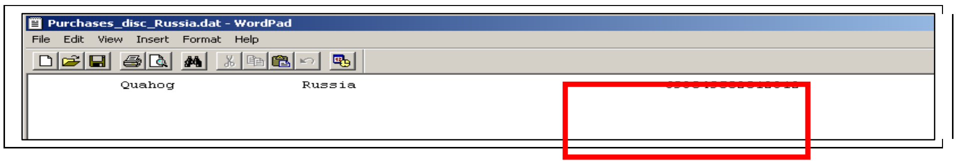
Data: C:\student\MessageModeling\discriminators\Purchases_disc_Russia.dat

Parsed Input

Characters

```
1 A0012345678Griffin              Peter               Pawtucket I
```

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

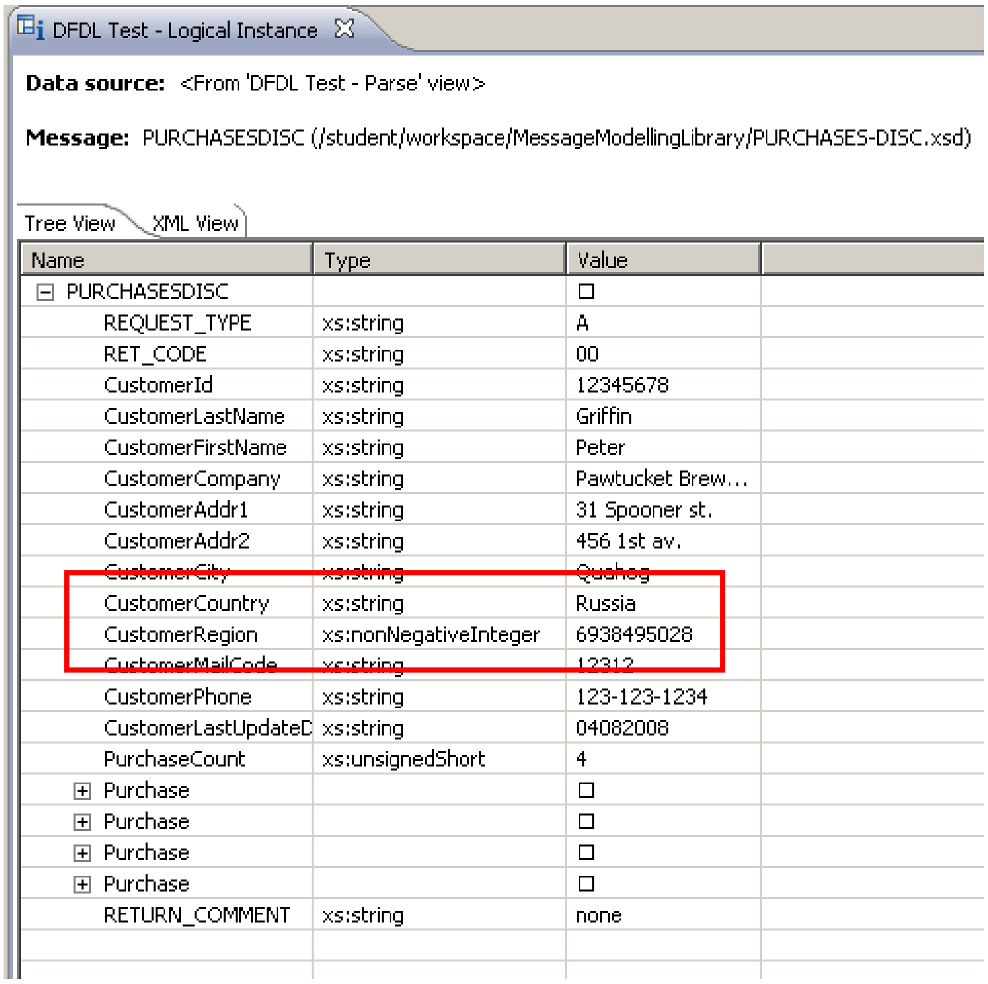

Purchases_disc_Russia.dat - WordPad
File Edit View Insert Format Help

```
            Quahog              Russia                        [illegible]
```

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.



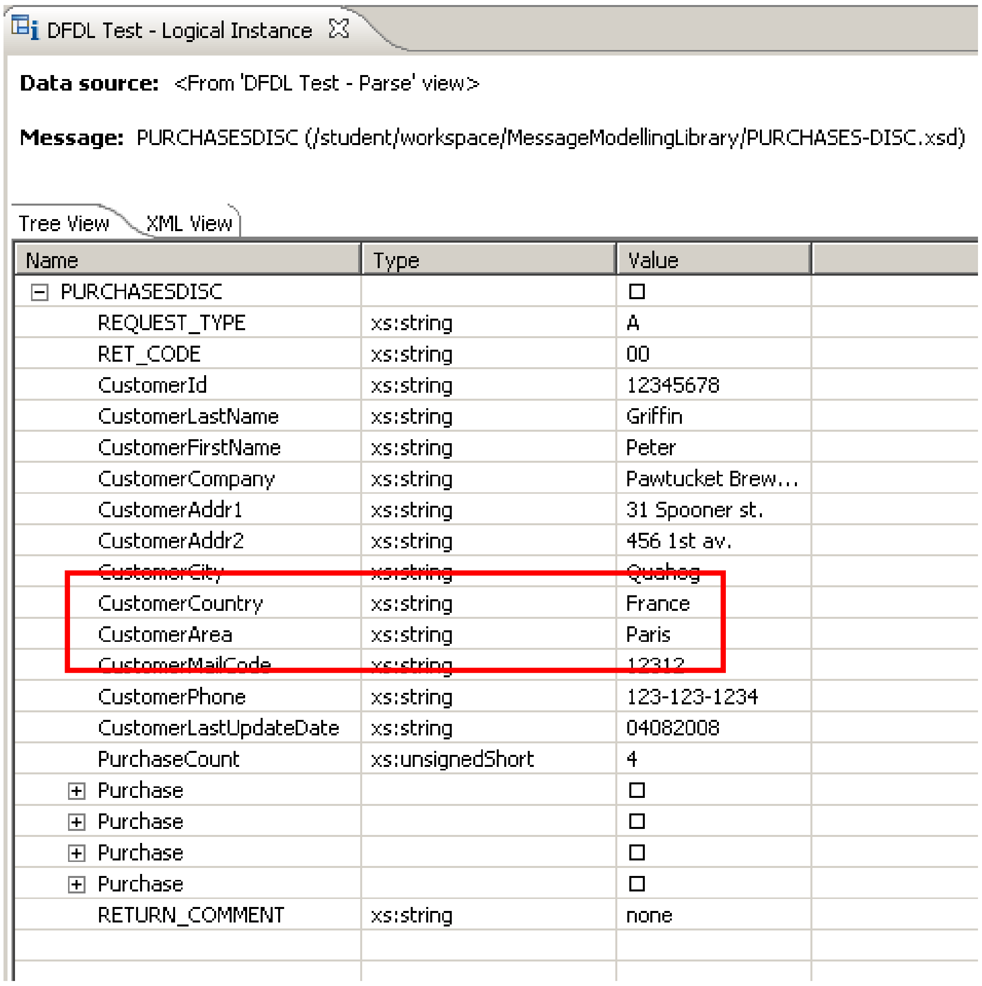
Data source: <From 'DFDL Test - Parse' view>

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

Name	Type	Value
⊟ PURCHASESDISC		☐
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		☐
⊞ Purchase		☐
⊞ Purchase		☐
⊞ Purchase		☐
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)

Name	Type	Value
PURCHASESDISC		☐
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		☐
Purchase		☐
Purchase		☐
Purchase		☐
RETURN_COMMENT	xs:string	none

This concludes the Message Modeling Discriminators lab.