16L92

IBM Integration Bus

Message Modeling with DFDL

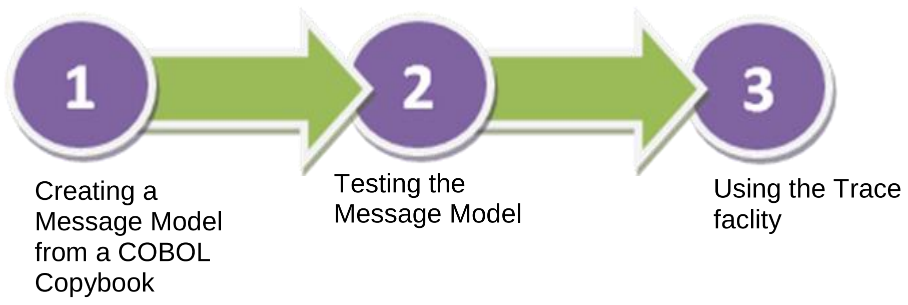
Lab 2
Modeling fixed-length data using a COBOL copybook

June 2015

Hands-on lab built at product
Version 10.0.0.0

1. Introduction

In this lab, you will create a Message Model from a COBOL Copybook. Then you will test parse it against a valid data file and a malformed data file. In this last part you will be able to take a look at the trace facility.


1
2
3
Creating a Message Model from a COBOL Copybook
Testing the Message Model
Using the Trace faclity

This lab should be done after Lab 1, Message Modelling with CSV files.

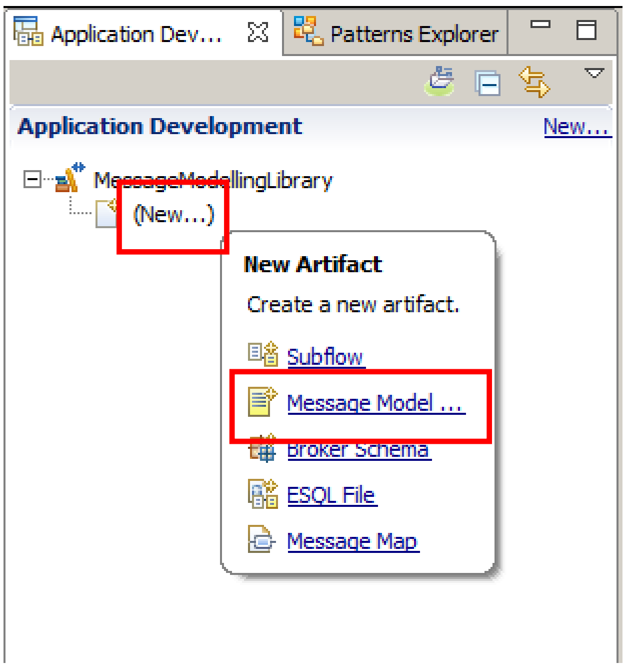
2. Creating a Message Model from a COBOL Copybook

This lab shows you how to create a Message Model based on a fixed length COBOL Copybook format. To do that, you will use the message model wizard taking a .cpy file as input.

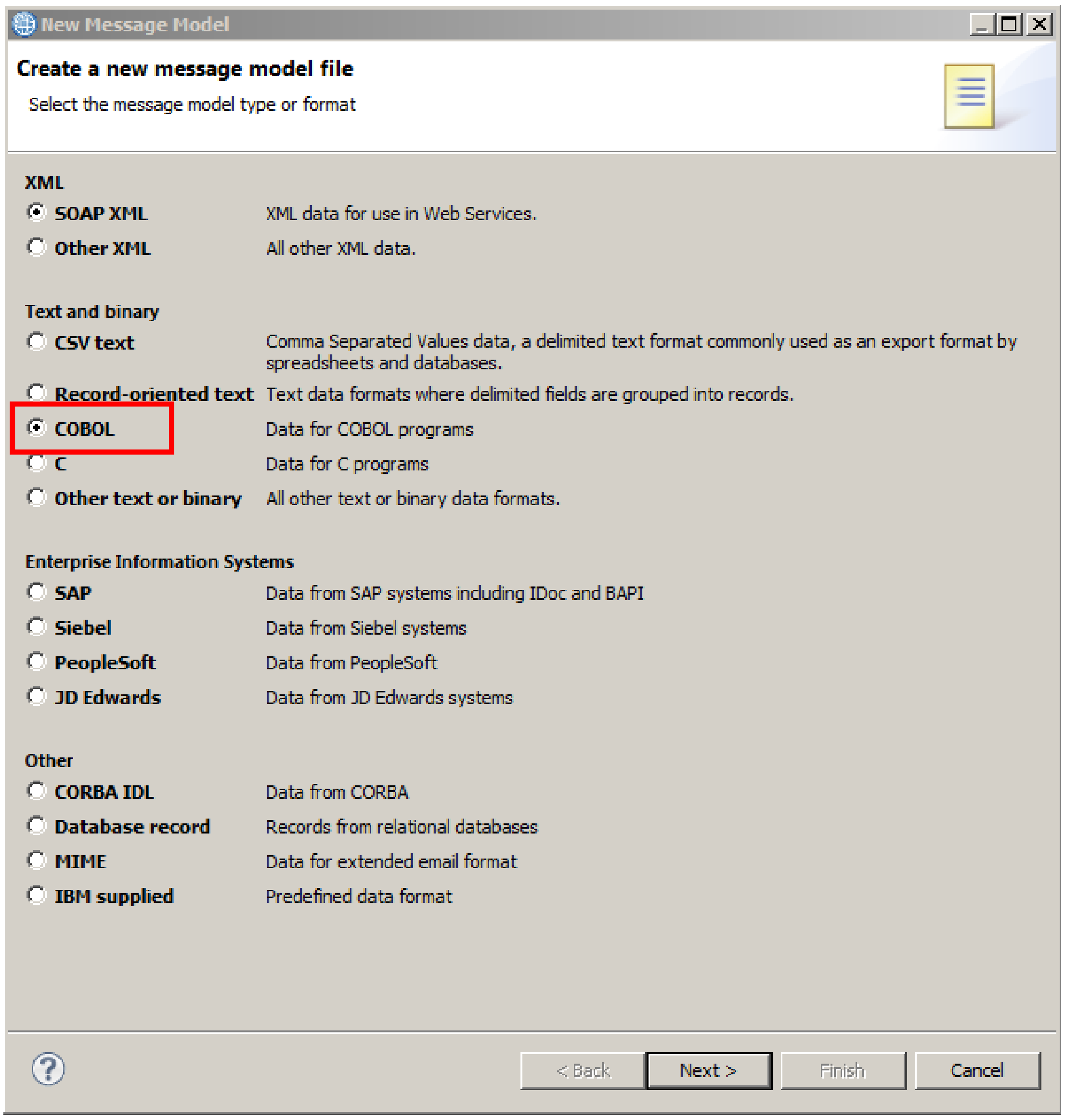
1. The first lab in this series created a CSV Message Model, so you should already have a Library created for this purpose. This lab uses the library called MessageModellingLibrary.

   (If you didn't do that lab, create a new library now called MessageModellingLibrary).

   In this library, click "New…", and select Message Model.



2. In the Message Model wizard, select COBOL and click Next.

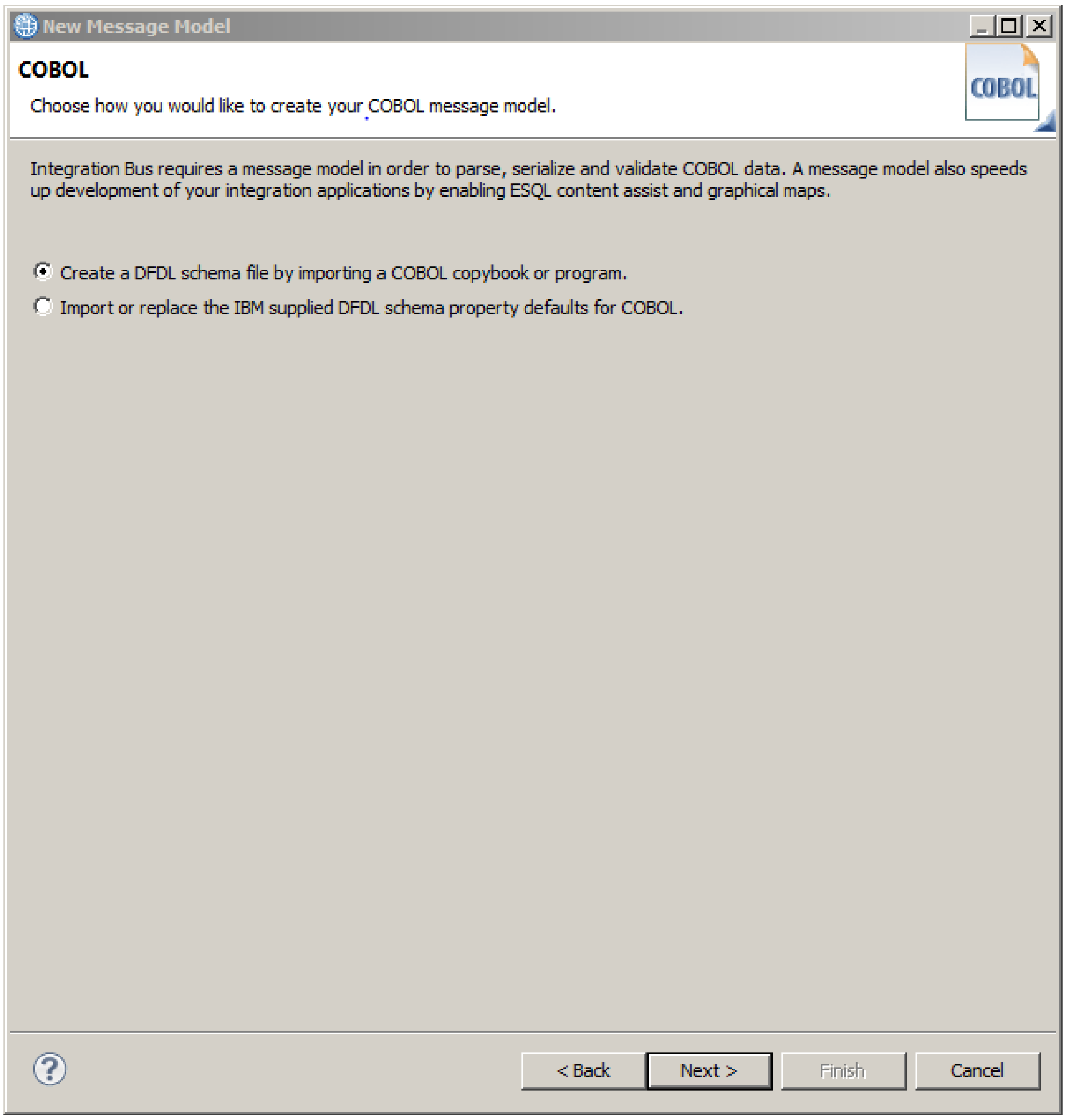


Create a new message model file

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
( ) Siebel	Data from Siebel systems
( ) PeopleSoft	Data from PeopleSoft
( ) JD Edwards	Data from JD Edwards systems
Other	
( ) CORBA IDL	Data from CORBA
( ) Database record	Records from relational databases
( ) MIME	Data for extended email format
( ) IBM supplied	Predefined data format

3. Leave the default option selected, and click Next.



COBOL

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

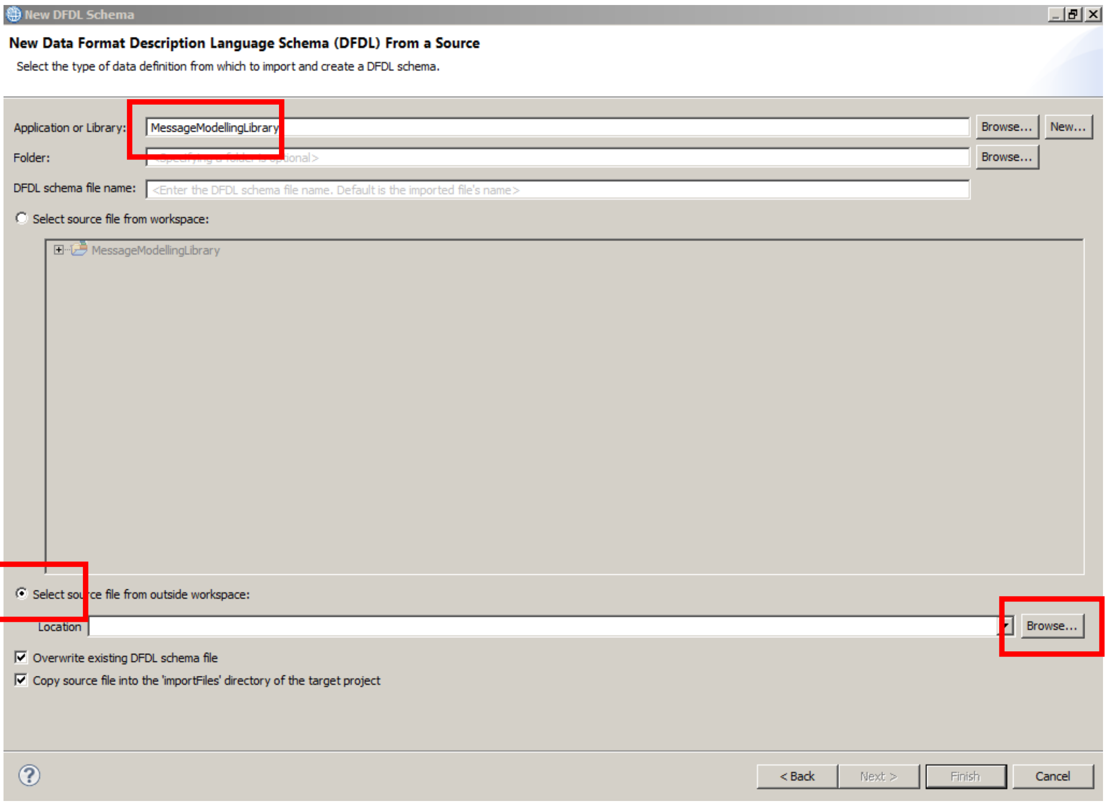
Integration Bus requires a message model in order to parse, serialize and validate COBOL data. A message model also speeds up development of your integration applications by enabling ESQL content assist and graphical maps.

(●) 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. Set the “Application or Library” to MessageModellingLibrary, by using the Browse button.

   Select "Select source file from outside workspace". Click the Browse button.


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 schema file name. Default is the imported file's name>

( ) Select source file from workspace:

MessageModellingLibrary

(•) Select source file from outside workspace:

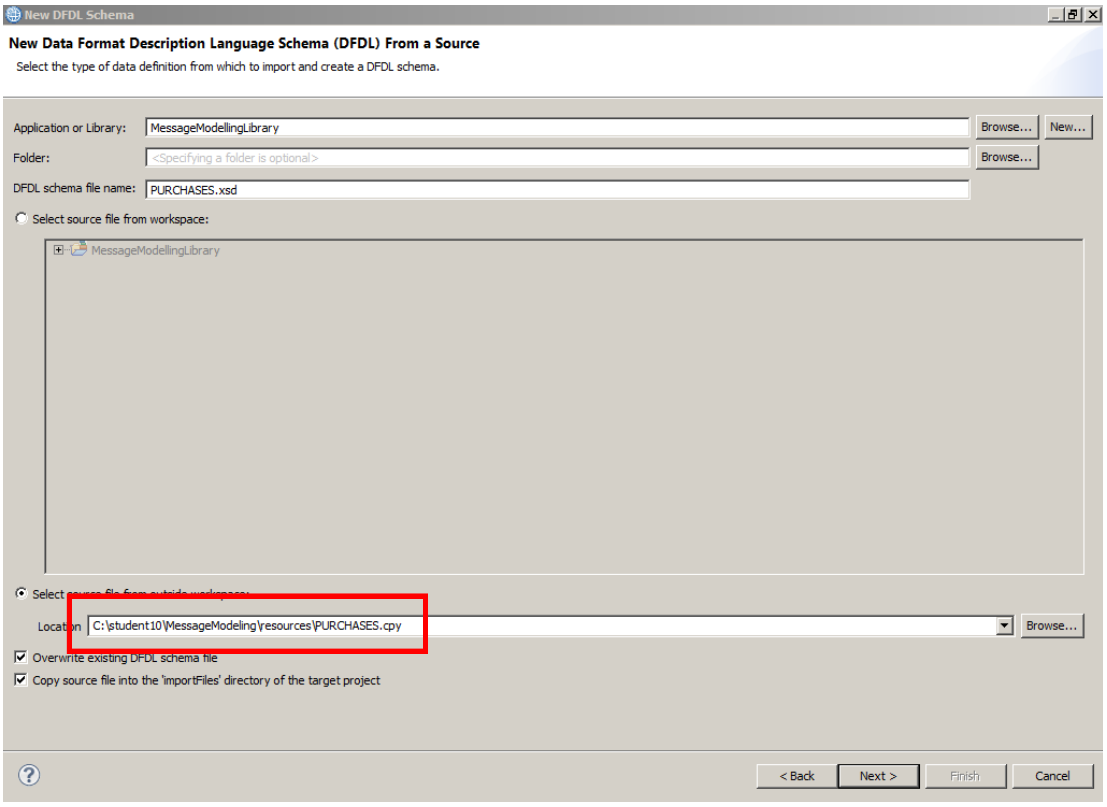
Location

[✓] Overwrite existing DFDL schema file

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

5. Browse in C:\student10\MessageModeling\resources\ and select the file "PURCHASES.cpy".

   Click 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.xsd

Select source file from workspace:

Select source file from outside workspace:

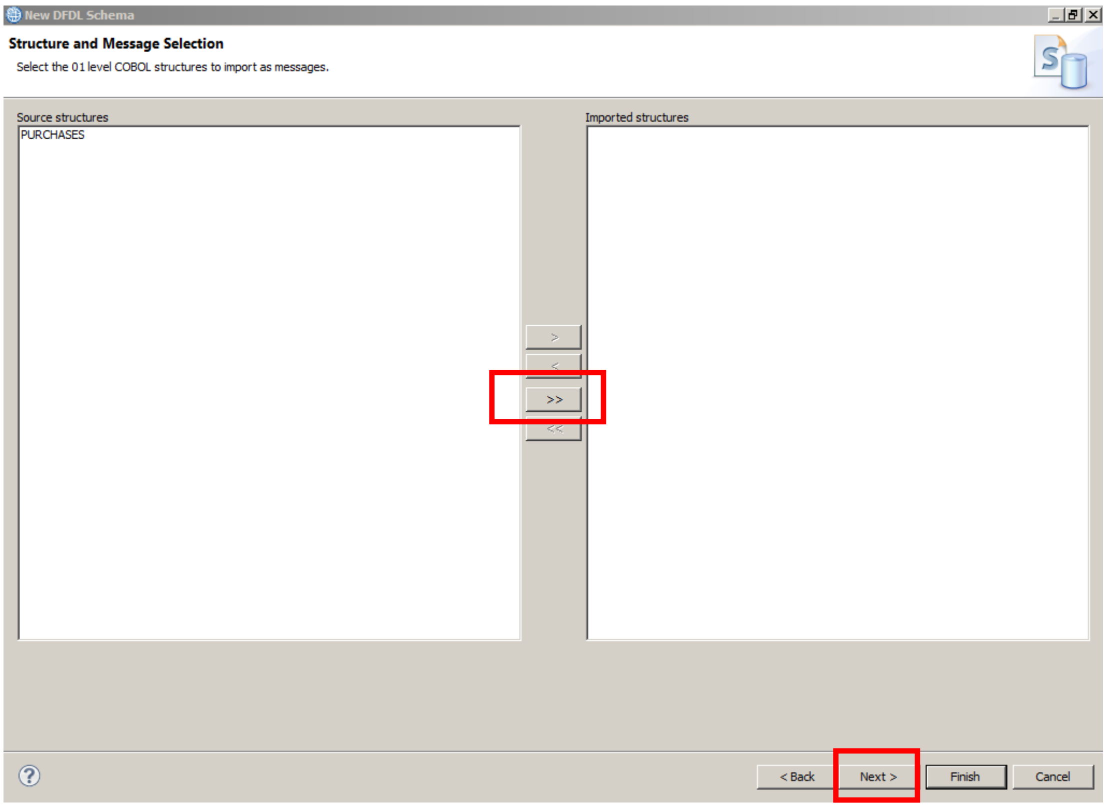
Location: C:\student10\MessageModeling\resources\PURCHASES.cpy

[✓] Overwrite existing DFDL schema file

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

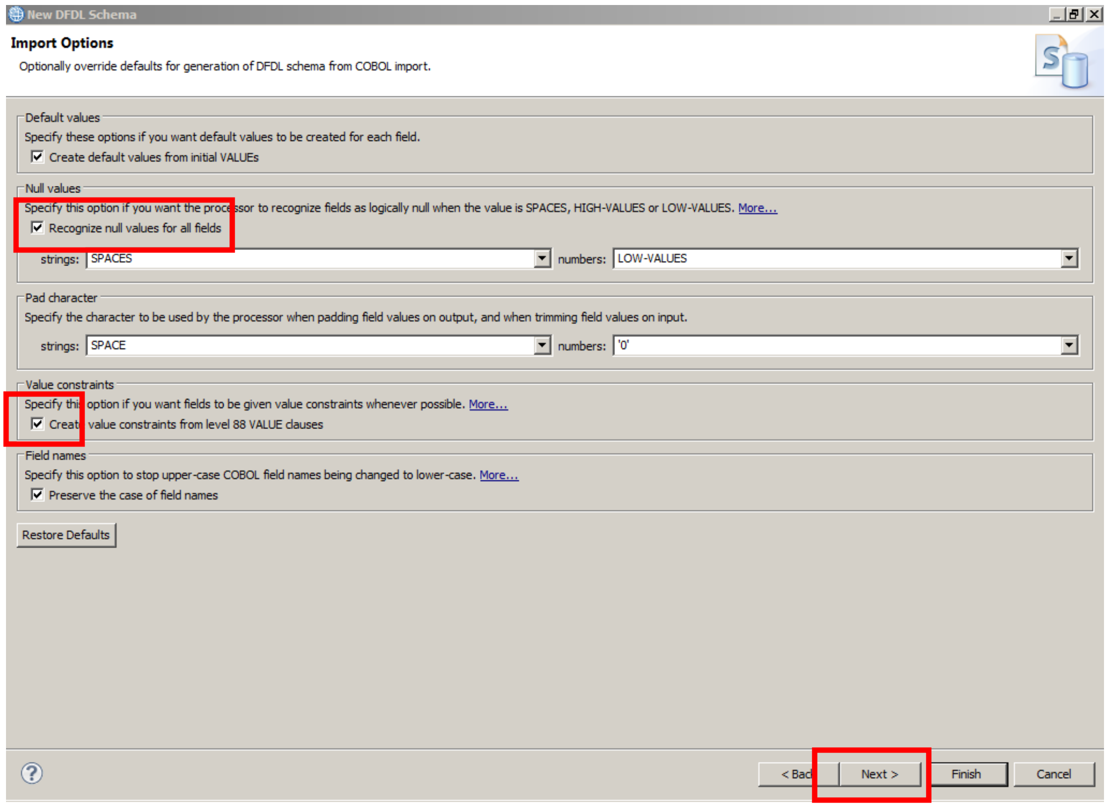
6. Click on the ">>" button to select all found objects (just one in this case) and click Next.

   Do not click Finish.


New DFDL Schema
Structure and Message Selection
Select the 01 level COBOL structures to import as messages.
Source structures
PURCHASES
Imported structures
>
<
>>
<<
< Back
Next >
Finish
Cancel

7. Leave most of the default values, but select “Recognize null values for all fields” and “Create value constraints from level 88 VALUE clauses”.

   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'

Value constraints

Specify this option if you want fields to be given value constraints whenever possible. More...

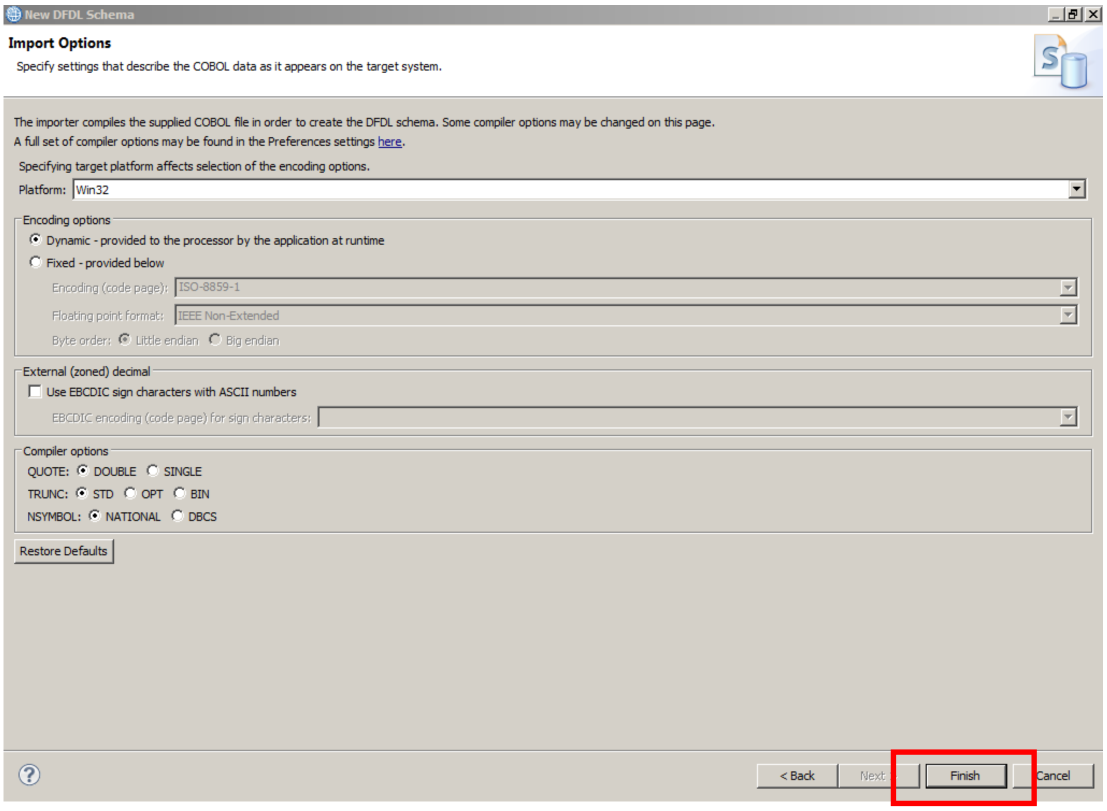
☑ Create value constraints from level 88 VALUE clauses

Field names

Specify this option to stop upper-case COBOL field names being changed to lower-case. More...

☑ Preserve the case of field names

8. Leave all the defaults values, and 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:

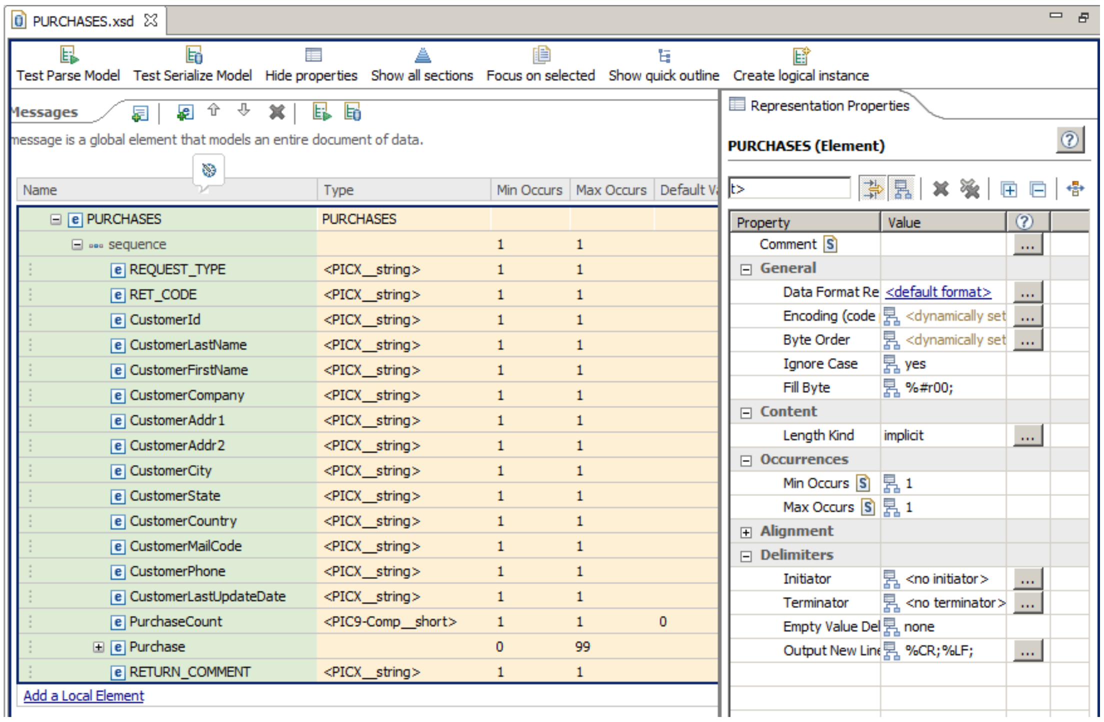
Compiler options

QUOTE: [✓ DOUBLE] SINGLE

TRUNC: [✓ STD] OPT BIN

NSYMBOL: [✓ NATIONAL] DBCS

9. The DFDL editor opens with the newly created DFDL message model called PURCHASES.xsd.



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

Name	Type	Min Occurs	Max Occurs	Default V
PURCHASES	PURCHASES			
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	
CustomerState	<PICX__string>	1	1	
CustomerCountry	<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	

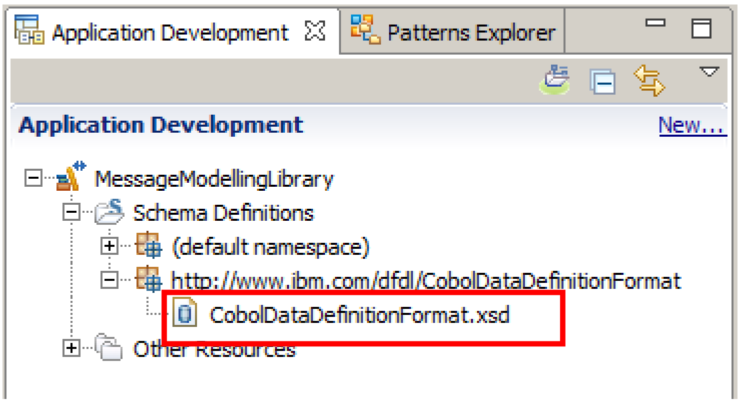
Add a Local Element

PURCHASES (Element)

Property	Value
Comment	
General	
Data Format Re	<default format>
Encoding (code	<dynamically set
Byte Order	<dynamically set
Ignore Case	yes
Fill Byte	%#r00;
Content	
Length Kind	implicit
Occurrences	
Min Occurs	1
Max Occurs	1
Alignment	
Delimiters	
Initiator	<no initiator>
Terminator	<no terminator>
Empty Value Del	none
Output New Line	%CR;%LF;

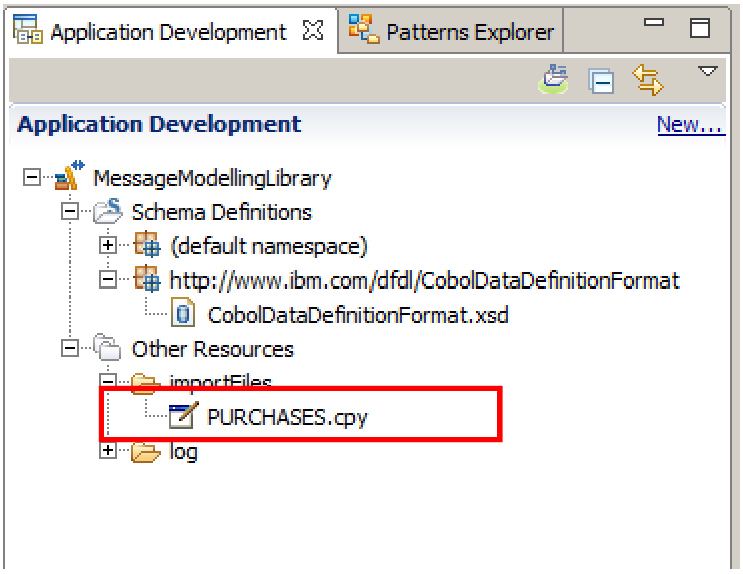
10. Notice the wizard automatically added a file called "CobolDataDefinitionFormat.xsd" under the Schema Definitions in MessageModellingLibrary.

This file is referenced by PURCHASES.xsd as a schema import, and it contains COBOL-specific defaults for all the DFDL properties, and some pre-defined simple types.

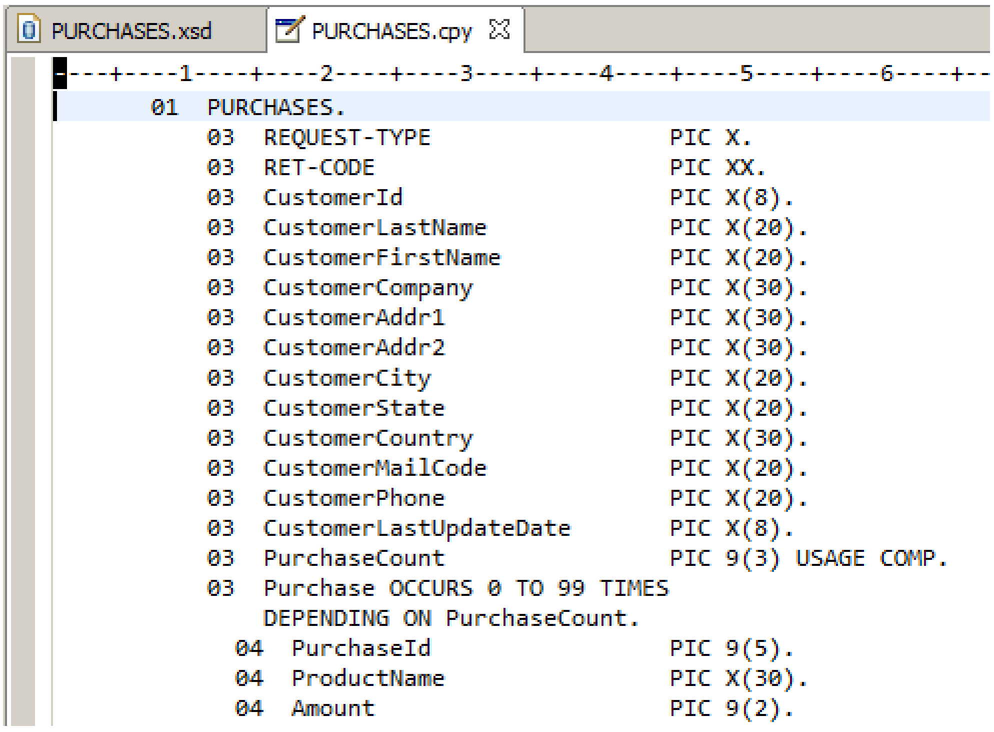


11. Expand the "Other Resources" folder under the MessageModelingLibrary library.

    Expand the "importFiles" folder and you will see the PURCHASES.cpy file that the wizard has automatically imported.



12. Double-click PURCHASES.cpy to open it in the editor.

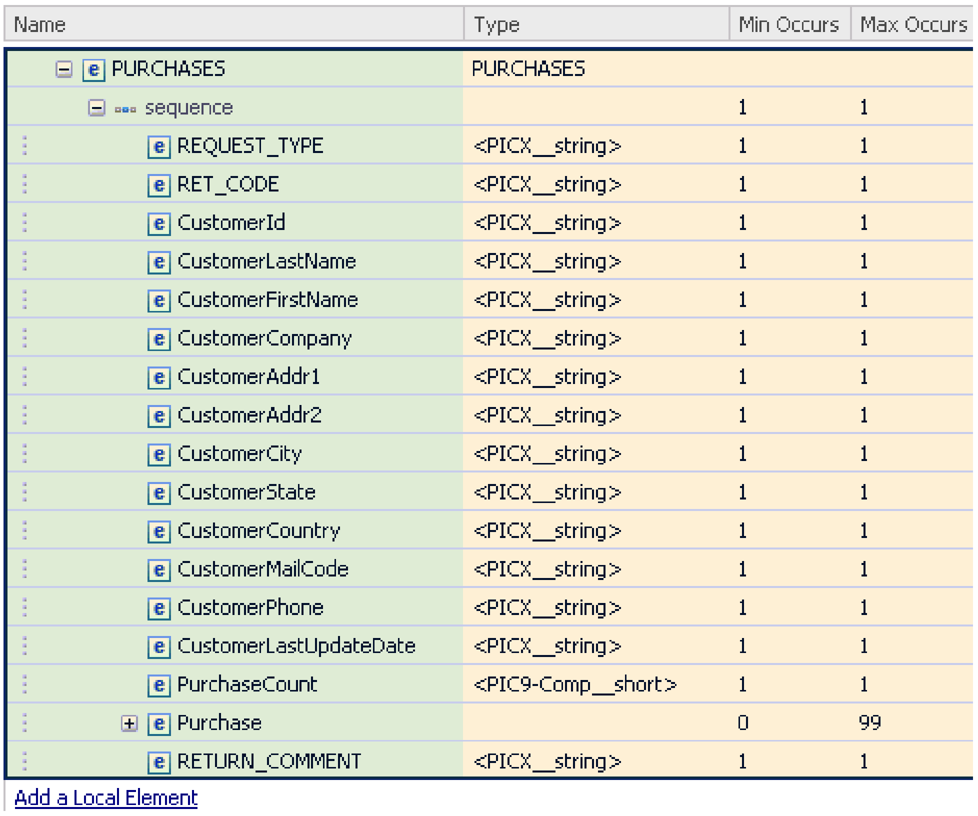


```
----+----1----+----2----+----3----+----4----+----5----+----6----+--
       01  PURCHASES.
           03  REQUEST-TYPE                 PIC X.
           03  RET-CODE                     PIC XX.
           03  CustomerId                   PIC X(8).
           03  CustomerLastName             PIC X(20).
           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  CustomerState                PIC X(20).
           03  CustomerCountry              PIC X(30).
           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).
             04  Amount                     PIC 9(2).
```

This is a simple copybook with:

1. 14 string fields
2. PurchaseCount: binary field with the number of the Purchase structure occurrences
3. Purchase: Repeating structure
    1. PurchaseId, Amount: numeric fields.
    2. Price: numeric field with 2 decimal places.

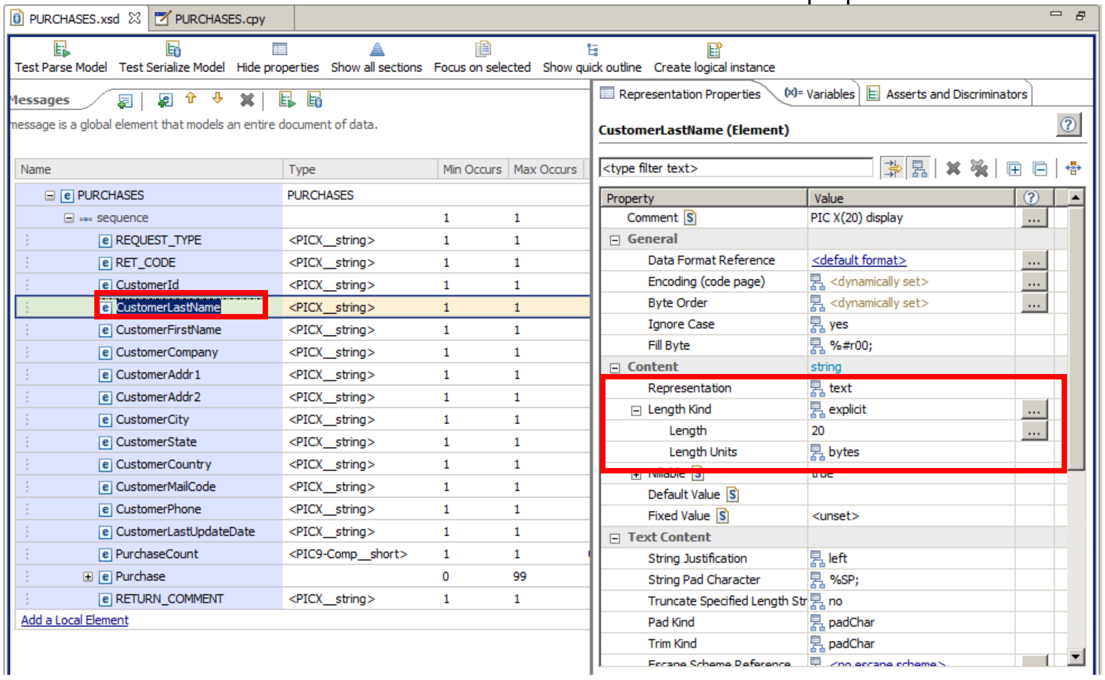
13. Switch to the DFDL editor. For the PURCHASES.xsd This shows the string fields, defined as "PICX_string" by the import wizard:



Name	Type	Min Occurs	Max Occurs
PURCHASES	PURCHASES		
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
CustomerState	<PICX__string>	1	1
CustomerCountry	<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
Purchase		0	99
RETURN_COMMENT	<PICX__string>	1	1

Add a Local Element

14. In the DFDL Editor click on the "CustomerLastName" field to see its properties:



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

Name	Type	Min Occurs	Max Occurs
PURCHASES	PURCHASES		
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
CustomerState	<PICX__string>	1	1
CustomerCountry	<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
Purchase		0	99
RETURN_COMMENT	<PICX__string>	1	1

Add a Local Element

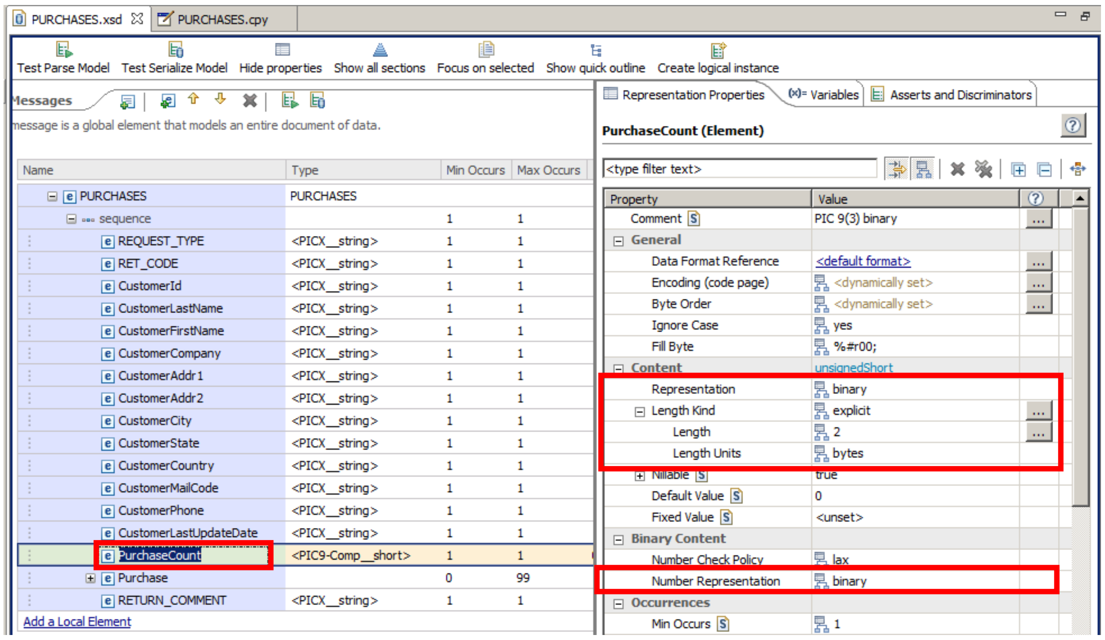
CustomerLastName (Element)

<type filter text>

Property	Value
Comment	PIC X(20) display
General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	yes
Fill Byte	%#r00;
Content	string
Representation	text
Length Kind	explicit
Length	20
Length Units	bytes
Nillable	true
Default Value	
Fixed Value	<unset>
Text Content	
String Justification	left
String Pad Character	%SP;
Truncate Specified Length Str	no
Pad Kind	padChar
Trim Kind	padChar
Escape Scheme Reference	<no escape scheme>

In the properties view, look for the "Content" section. Note that the field was modeled as "text" representation, with a fixed (explicit) Length of 20 bytes, because the cpy file defined it as a "PIC X(20)"

15. In the DFDL Editor, click on the PurchaseCount field to see its properties:



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

Name	Type	Min Occurs	Max Occurs
PURCHASES	PURCHASES		
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
CustomerState	<PICX__string>	1	1
CustomerCountry	<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
Purchase		0	99
RETURN_COMMENT	<PICX__string>	1	1

PurchaseCount (Element)

Property	Value
Comment	PIC 9(3) binary
General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	yes
Fill Byte	%#r00;
Content	unsignedShort
Representation	binary
Length Kind	explicit
Length	2
Length Units	bytes
Nillable	true
Default Value	0
Fixed Value	<unset>
Binary Content	
Number Check Policy	lax
Number Representation	binary
Occurrences	
Min Occurs	1

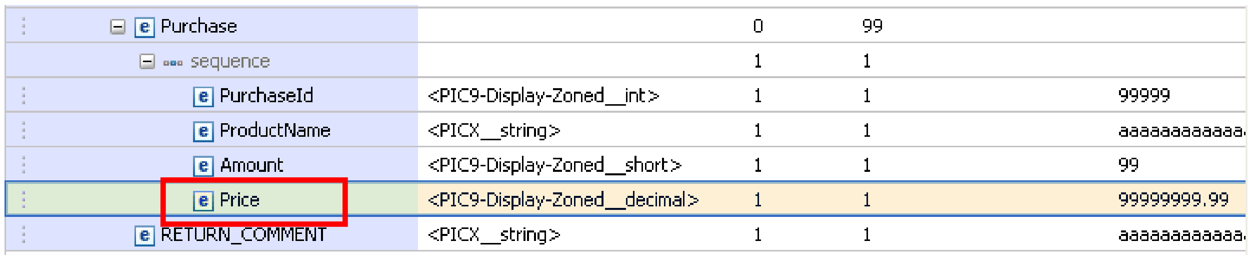
This field, which was defined as binary in the copybook file ("PIC 9(3) USAGE COMP"), was created as "PIC9_Comp_short" by the Import wizard.

You can see the details of this field in the properties view, where its length is set to "2", its Length Units to "bytes" and its representation to "binary".

Also, in the "Binary Content" section, its Binary Number Representation is set to binary. This property can take 4 different values:

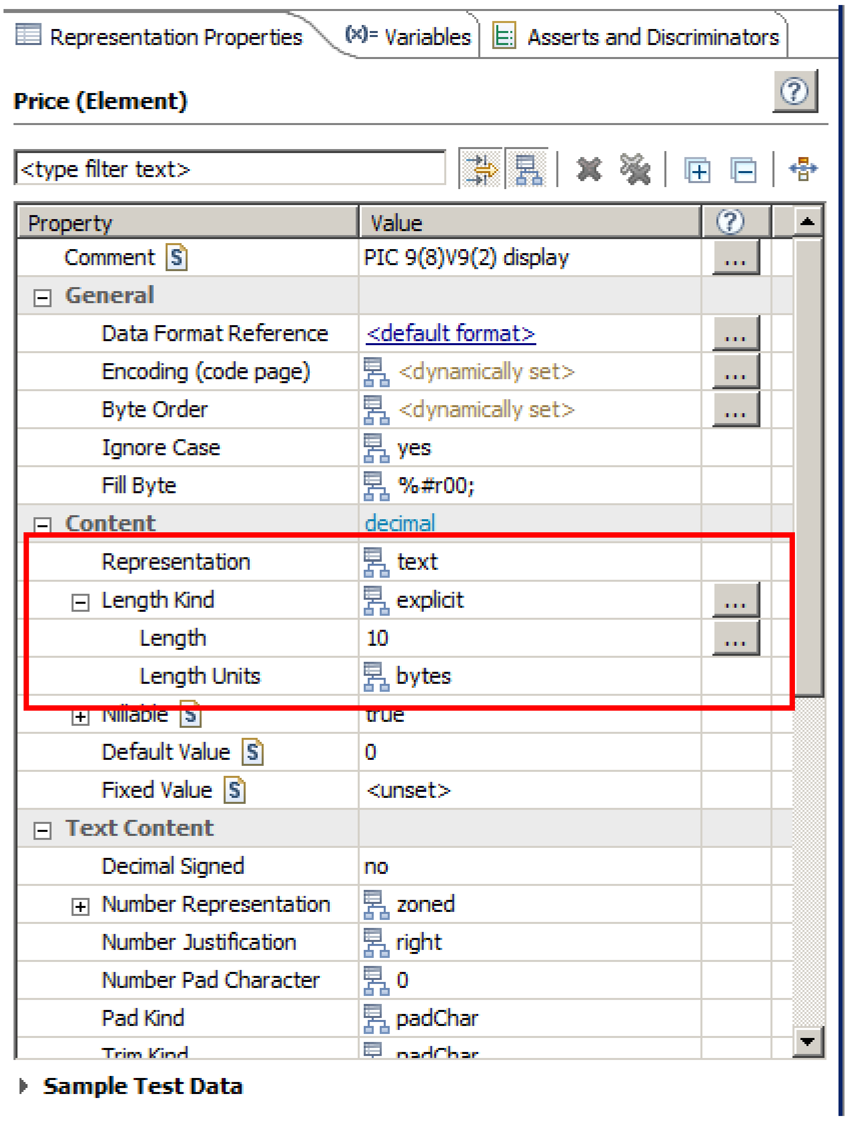
- packed: represented as a packed decimal. Each byte contains 2 decimal digits except for the least significant byte, which contains a sign in the least significant nibble
- bcd: represented as a binary coded decimal with 2 digits per byte.
- binary: represented as 2' complement for signed types and unsigned binary for unsigned types.
- ibm4690Packed: used by the IBM 4690 retail store devices

16. Now click on the Price field, in the Purchase structure.



Purchase		0	99	
sequence		1	1	
PurchaseId	<PIC9-Display-Zoned__int>	1	1	99999
ProductName	<PICX__string>	1	1	aaaaaaaaaaaa
Amount	<PIC9-Display-Zoned__short>	1	1	99
Price	<PIC9-Display-Zoned__decimal>	1	1	99999999.99
RETURN_COMMENT	<PICX__string>	1	1	aaaaaaaaaaaa

17. In the properties view, look at the "Content" section.

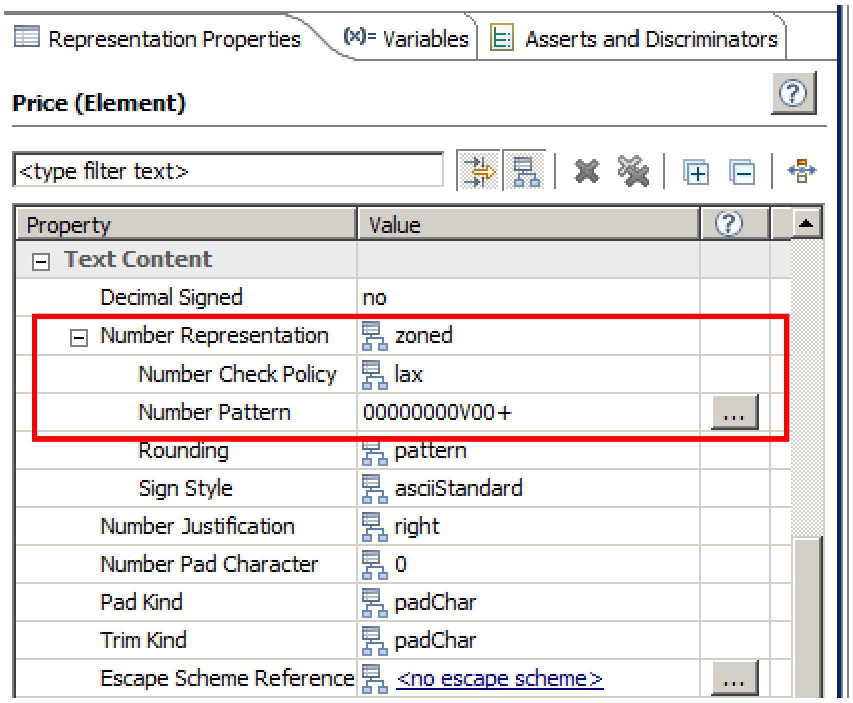


Price (Element)

Property	Value
Comment	PIC 9(8)V9(2) display
General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	yes
Fill Byte	%#r00;
Content	decimal
Representation	text
Length Kind	explicit
Length	10
Length Units	bytes
Nillable	true
Default Value	0
Fixed Value	<unset>
Text Content	
Decimal Signed	no
Number Representation	zoned
Number Justification	right
Number Pad Character	0
Pad Kind	padChar
Trim Kind	padChar

Note that it is defined as a decimal field, with text representation and a length of 10 bytes (8 integers and 2 decimal places).

18. Look at the "Text Content" section of the properties view.



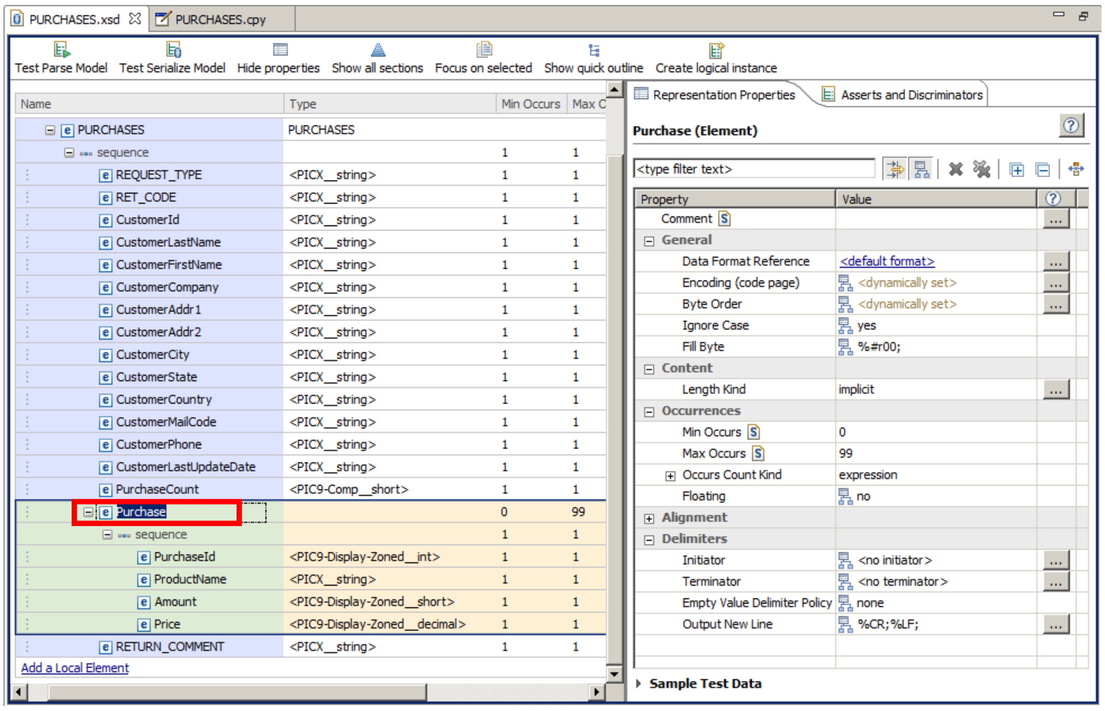
Price (Element)

Property	Value
Text Content	
Decimal Signed	no
Number Representation	zoned
Number Check Policy	lax
Number Pattern	00000000V00+
Rounding	pattern
Sign Style	asciiStandard
Number Justification	right
Number Pad Character	0
Pad Kind	padChar
Trim Kind	padChar
Escape Scheme Reference	<no escape scheme>

Note that the "Number Representation" is defined as "zoned", with a pattern of 8 integer numbers and 2 decimal places.

The letter “V” in the Number Pattern is an implied decimal point (common in COBOL copybooks).

19. Click on the "Purchase" element to open its properties.



PURCHASES.xsd | PURCHASES.cpy

Test Parse Model | Test Serialize Model | Hide properties | Show all sections | Focus on selected | Show quick outline | Create logical instance

Name	Type	Min Occurs	Max C
PURCHASES	PURCHASES		
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
CustomerState	<PICX__string>	1	1
CustomerCountry	<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
Purchase		0	99
sequence		1	1
PurchaseId	<PIC9-Display-Zoned__int>	1	1
ProductName	<PICX__string>	1	1
Amount	<PIC9-Display-Zoned__short>	1	1
Price	<PIC9-Display-Zoned__decimal>	1	1
RETURN_COMMENT	<PICX__string>	1	1

Add a Local Element

Representation Properties | Asserts and Discriminators

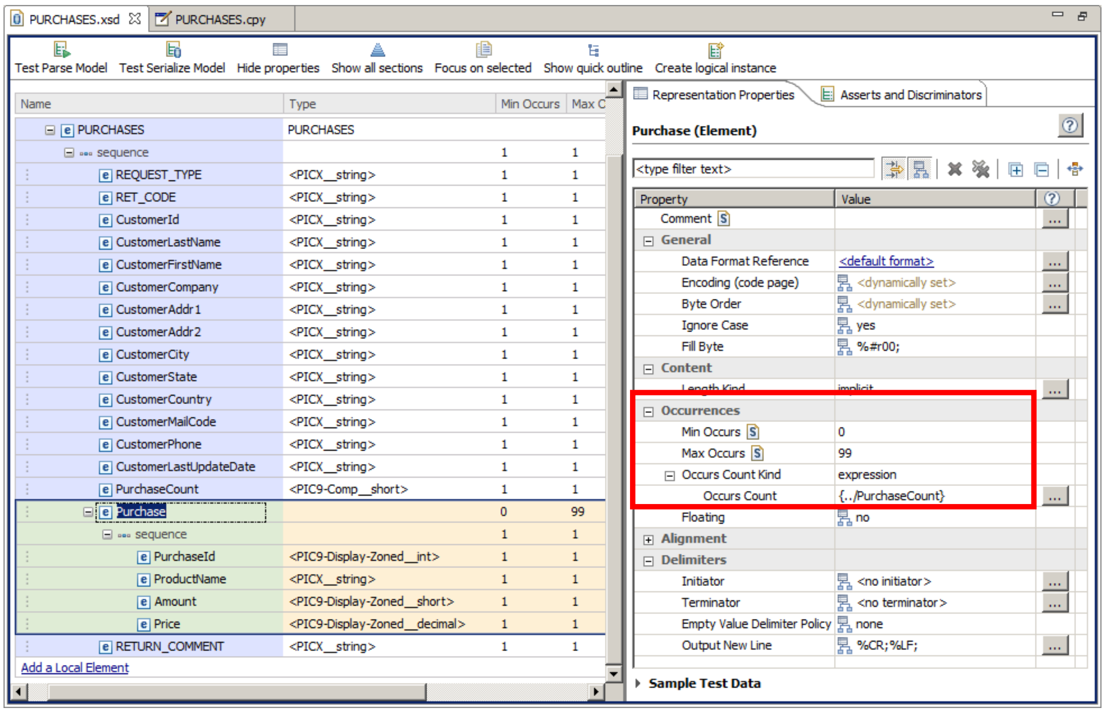
Purchase (Element)

<type filter text>

Property	Value
Comment	
General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	yes
Fill Byte	%#r00;
Content	
Length Kind	implicit
Occurrences	
Min Occurs	0
Max Occurs	99
Occurs Count Kind	expression
Floating	no
Alignment	
Delimiters	
Initiator	<no initiator>
Terminator	<no terminator>
Empty Value Delimiter Policy	none
Output New Line	%CR;%LF;

Sample Test Data

20. Look for the "Occurrences" section inside the properties view, and expand the "Occurs Count Kind" property.



Name	Type	Min Occurs	Max Occurs
PURCHASES	PURCHASES		
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
CustomerState	<PICX__string>	1	1
CustomerCountry	<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
Purchase		0	99
sequence		1	1
PurchaseId	<PIC9-Display-Zoned__int>	1	1
ProductName	<PICX__string>	1	1
Amount	<PIC9-Display-Zoned__short>	1	1
Price	<PIC9-Display-Zoned__decimal>	1	1
RETURN_COMMENT	<PICX__string>	1	1

Purchase (Element)

Property	Value
Comment	
General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	yes
Fill Byte	%#r00;
Content	
Length Kind	implicit
Occurrences	
Min Occurs	0
Max Occurs	99
Occurs Count Kind	expression
Occurs Count	{../PurchaseCount}
Floating	no
Alignment	
Delimiters	
Initiator	<no initiator>
Terminator	<no terminator>
Empty Value Delimiter Policy	none
Output New Line	%CR;%LF;

This property, as defined by the DFDL specification, can take different values:

1. fixed: uses the "maxOccurs" property
2. expression: uses the value defined by the expression in "occursCount" property.
3. parsed: the number of occurrences is determined by normal speculative parsing.
4. implicit: uses “minOccurs” and “maxOccurs” properties with speculative parsing

In this case, the "OccursCountKind" property is set to "expression", and "occursCount" is set to point to the "PurchaseCount" element. This means that the number of occurrences of the "Purchase" repeating structure will be defined by the PurchaseCount element.

This was defined by the Import wizard to reflect the cpy file, which stated:

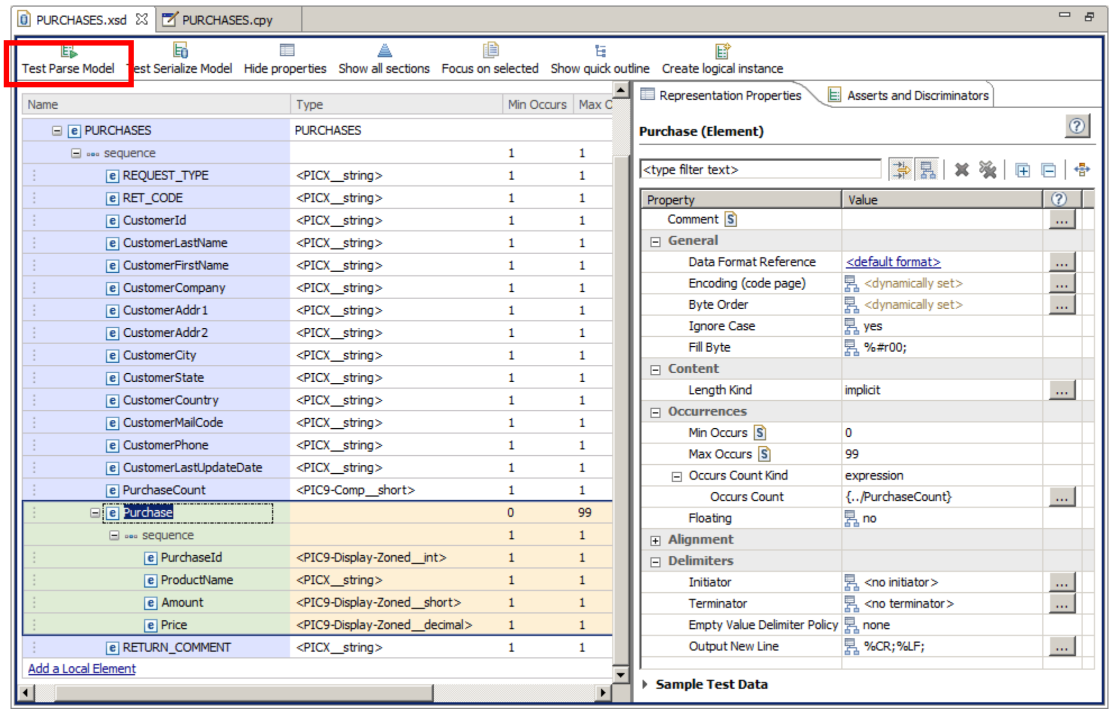
```
Purchase OCCURS 0 TO 99 TIMES DEPENDING ON PurchaseCount.
```

Notice also that the MinOccurs property is set to "0" and the MaxOccurs property is set to "99", as the cpy file stated.

21. Save your message model (PURCHASES.xsd) by pressing Ctrl+S, or File->Save.

3. Testing the Message Model

1. Now you will test that the message model correctly models the COBOL data. Click the "Test Parse Model" icon.

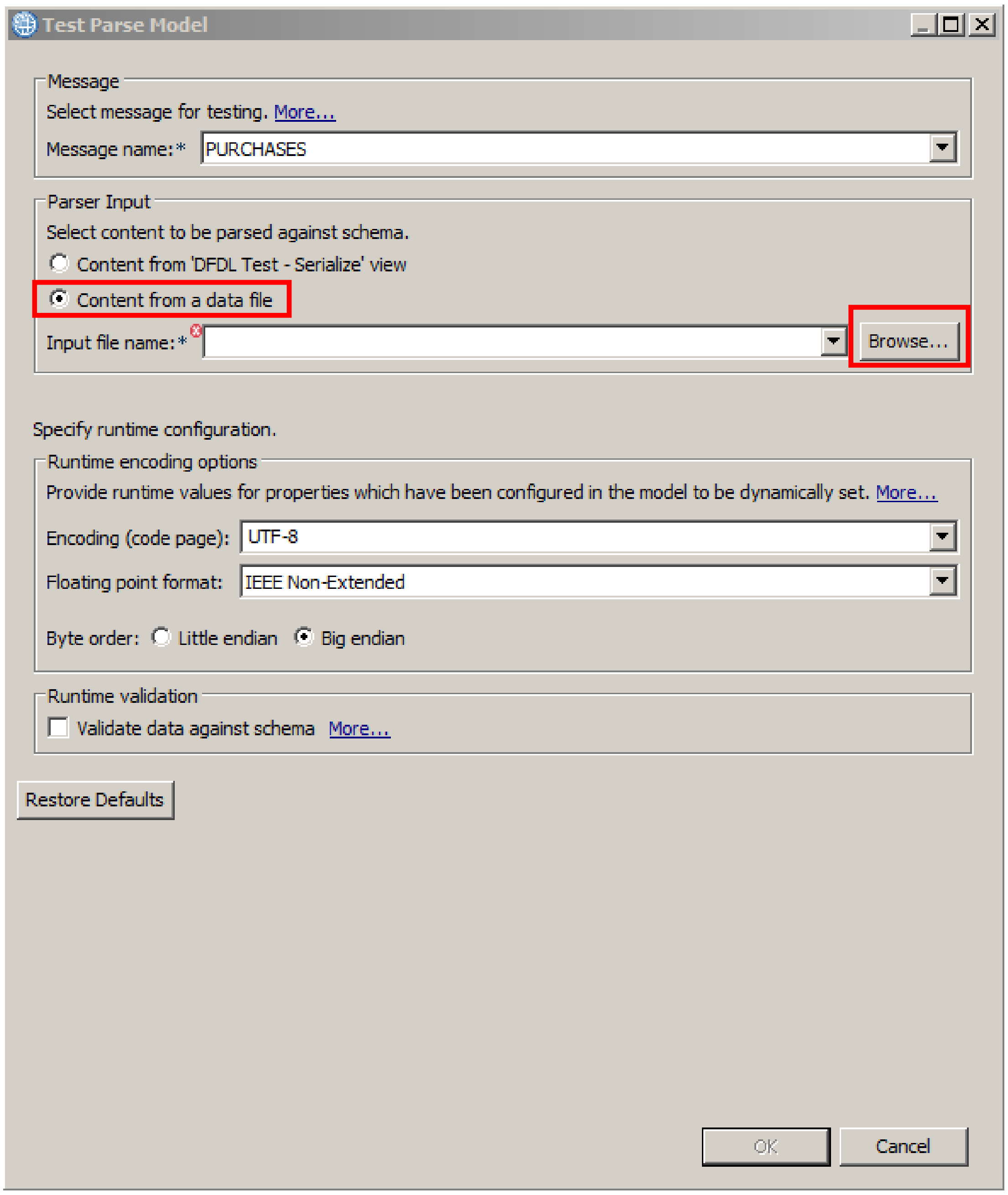


Name	Type	Min Occurs	Max O
PURCHASES	PURCHASES		
sequence		1	1
REQUEST_TYPE	<PICX__string>	1	1
RET_CODE	<PICX__string>	1	1
CustomerId	<PICX__string>	1	1
CustomerLastName	<PICX__string>	1	1
CustomerFirstName	<PICX__string>	1	1
CustomerCompany	<PICX__string>	1	1
CustomerAddr1	<PICX__string>	1	1
CustomerAddr2	<PICX__string>	1	1
CustomerCity	<PICX__string>	1	1
CustomerState	<PICX__string>	1	1
CustomerCountry	<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
Purchase		0	99
sequence		1	1
PurchaseId	<PIC9-Display-Zoned__int>	1	1
ProductName	<PICX__string>	1	1
Amount	<PIC9-Display-Zoned__short>	1	1
Price	<PIC9-Display-Zoned__decimal>	1	1
RETURN_COMMENT	<PICX__string>	1	1

Purchase (Element)

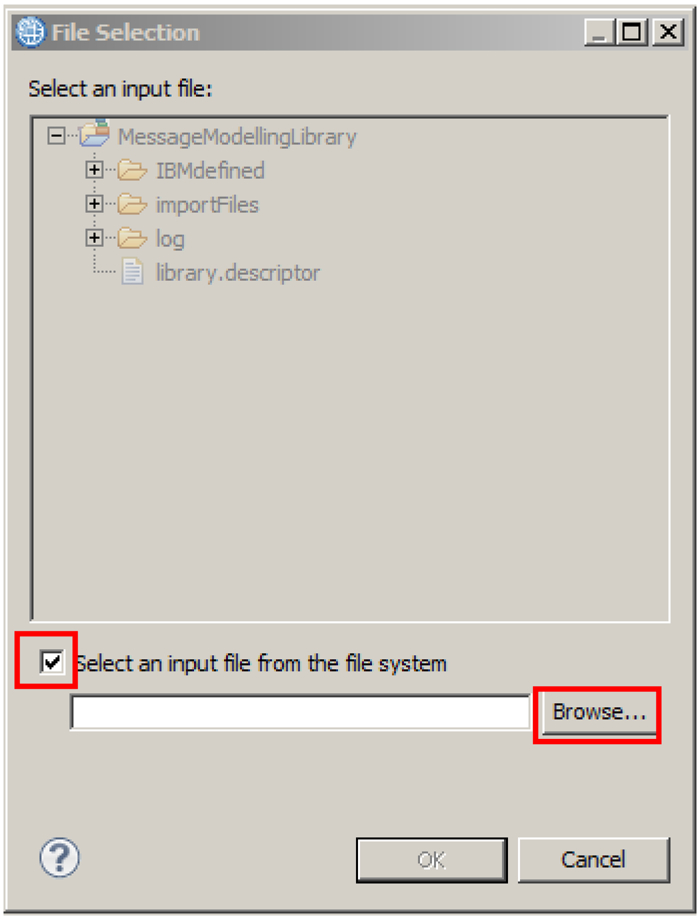
Property	Value
Comment	
General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	yes
Fill Byte	%#r00;
Content	
Length Kind	implicit
Occurrences	
Min Occurs	0
Max Occurs	99
Occurs Count Kind	expression
Occurs Count	{../PurchaseCount}
Floating	no
Alignment	
Delimiters	
Initiator	<no initiator>
Terminator	<no terminator>
Empty Value Delimiter Policy	none
Output New Line	%CR;%LF;

2. In the Parser Input section, select "Content from a data file" and click the Browse button.


Test Parse Model
Message
Select message for testing. More...
Message name:* PURCHASES
Parser Input
Select content to be parsed against schema.
Content from 'DFDL Test - Serialize' view
Content from a data file
Input file name:*
Browse...
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...
Restore Defaults
OK
Cancel

3. In the File Selection dialog, select the "Select an input file from the file system" option.

   Click on the Browse button.


File Selection

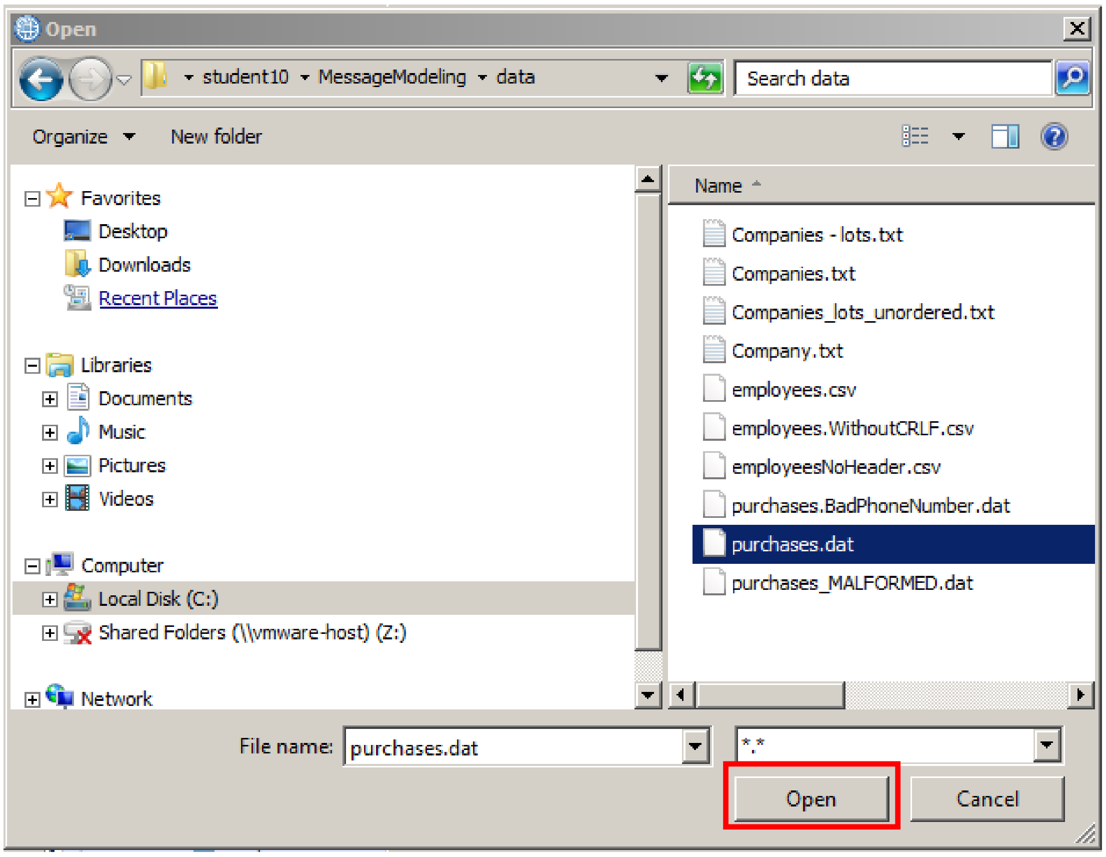
Select an input file:

- MessageModellingLibrary
  - IBMdefined
  - importFiles
  - log
  - library.descriptor

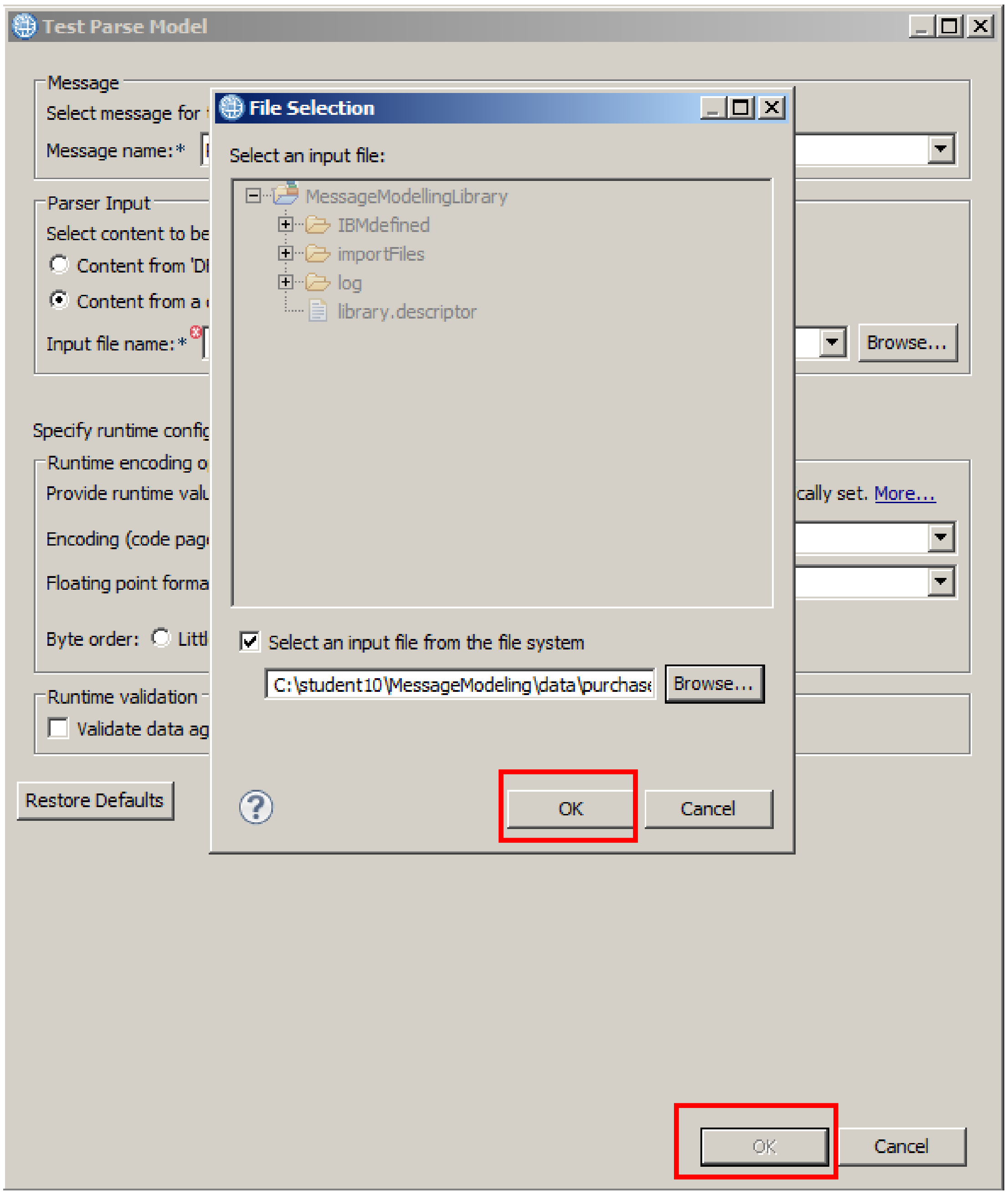
[✓] Select an input file from the file system

4. Navigate to "C:\student10\MessageModeling\data\" and select the "purchases.dat" file.

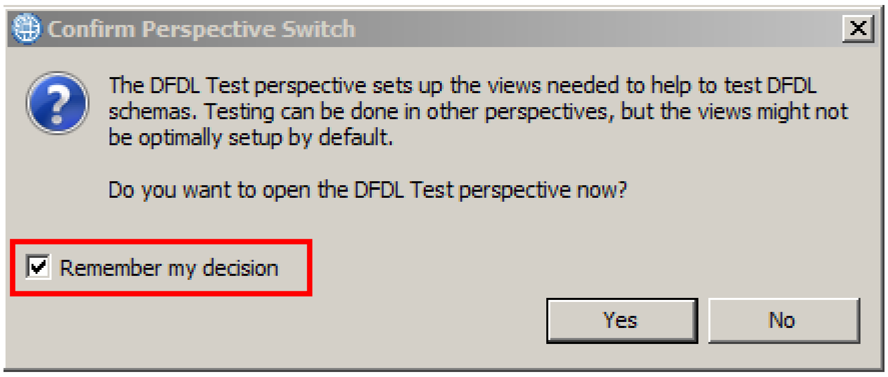
   Click the Open button.



5. Click OK on both windows.



6. Click on the checkbox "Remember my decision", and click Yes.



Confirm Perspective Switch

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.

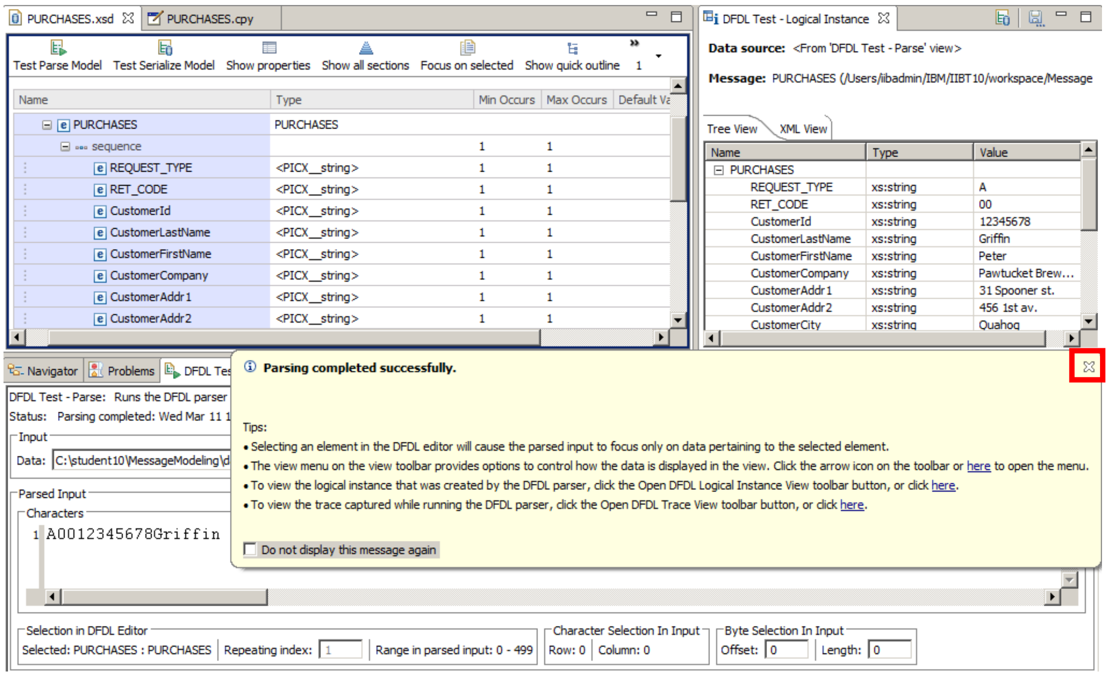
Do you want to open the DFDL Test perspective now?

[✓] Remember my decision

7. The DFDL Test perspective will open, with the Test Parse view in focus.

   A message balloon will appear, indicating the parsing was successful.

   Close it by clicking on the "x", or by clicking anywhere else in the workbench.



Name	Type	Min Occurs	Max Occurs	Default Va
PURCHASES	PURCHASES			
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	

DFDL Test - Logical Instance

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

Message: PURCHASES (/Users/iibadmin/IBM/IIBT10/workspace/Message

Name	Type	Value
PURCHASES		
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

DFDL Test - Parse: Runs the DFDL parser
Status: Parsing completed: Wed Mar 11 1

Input
Data: C:\student10\MessageModeling\d

Parsed Input
Characters
1 A0012345678Griffin

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.

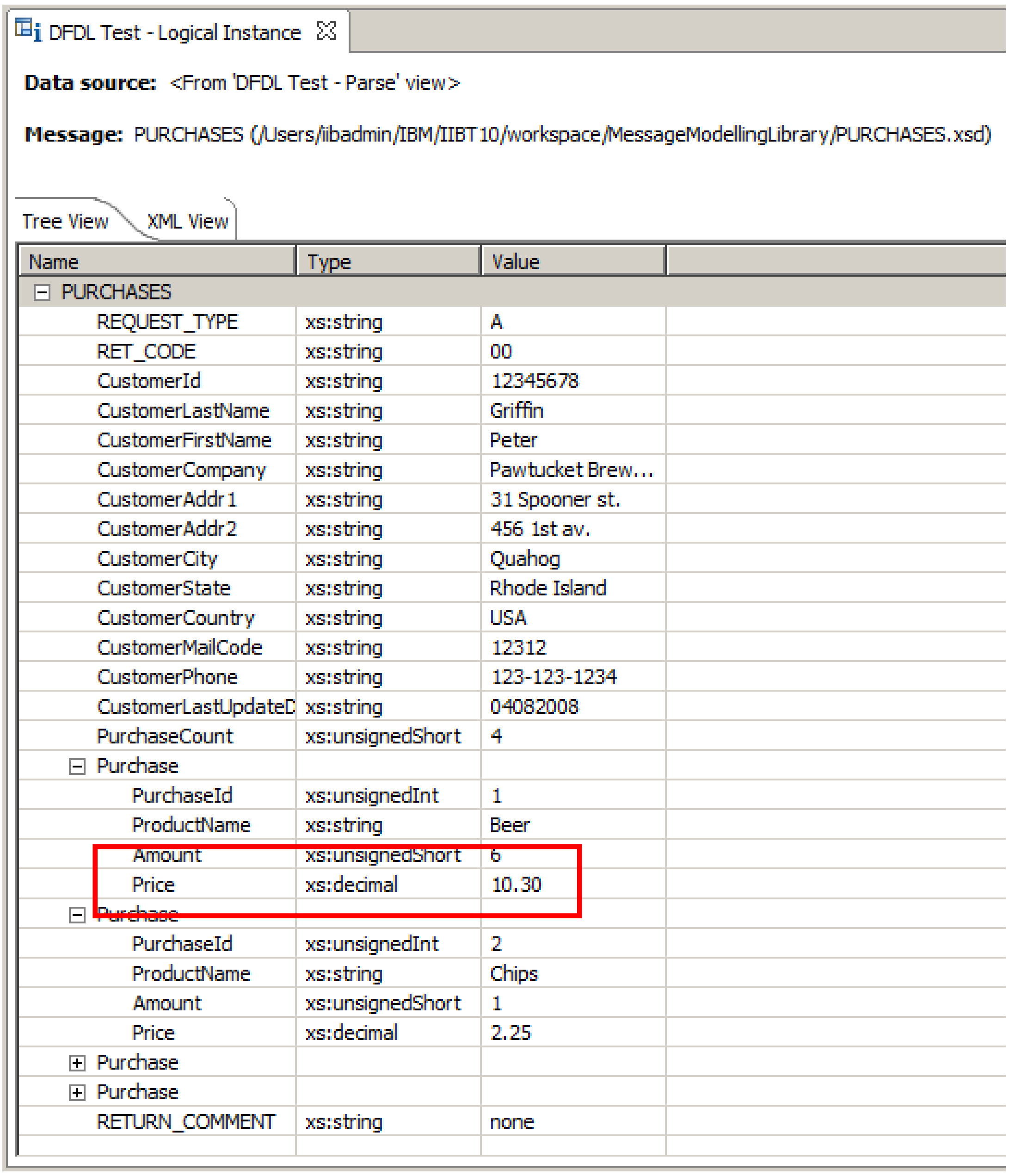
☐ Do not display this message again

Selection in DFDL Editor
Selected: PURCHASES : PURCHASES | Repeating index: 1 | Range in parsed input: 0 - 499

Character Selection In Input
Row: 0 | Column: 0

Byte Selection In Input
Offset: 0 | Length: 0

8. Inspect the "Test - Logical Instance" view. Navigate through the message tree parsed from the input file.



DFDL Test - Logical Instance

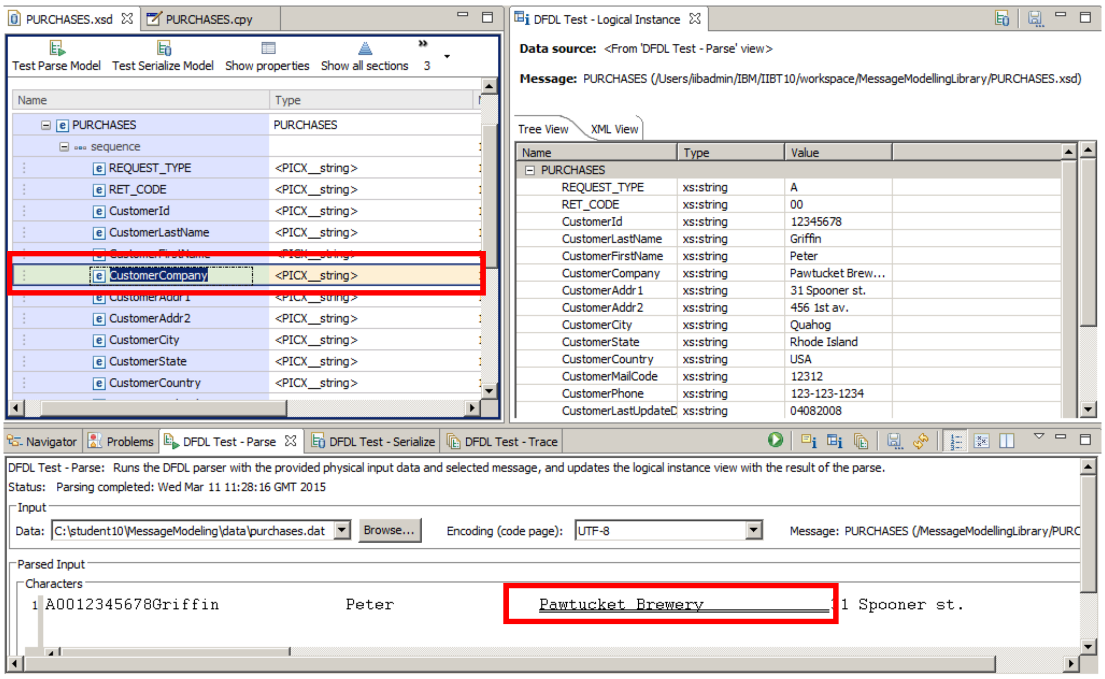
Data source: <From 'DFDL Test - Parse' view>

Message: PURCHASES (/Users/iibadmin/IBM/IIBT10/workspace/MessageModellingLibrary/PURCHASES.xsd)

Name	Type	Value
PURCHASES		
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
CustomerState	xs:string	Rhode Island
CustomerCountry	xs:string	USA
CustomerMailCode	xs:string	12312
CustomerPhone	xs:string	123-123-1234
CustomerLastUpdateD	xs:string	04082008
PurchaseCount	xs:unsignedShort	4
Purchase		
PurchaseId	xs:unsignedInt	1
ProductName	xs:string	Beer
Amount	xs:unsignedShort	6
Price	xs:decimal	10.30
Purchase		
PurchaseId	xs:unsignedInt	2
ProductName	xs:string	Chips
Amount	xs:unsignedShort	1
Price	xs:decimal	2.25
Purchase		
Purchase		
RETURN_COMMENT	xs:string	none

Note that the parser shows "10.30" (2 decimal places) because the COBOL field was defined as PIC 9(8)V99.

9. In the DFDL Editor, click on any element on the Message Model and you will see the relevant data underlined in the input text below:



Name	Type
PURCHASES	PURCHASES
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>
CustomerState	<PICX__string>
CustomerCountry	<PICX__string>

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

Message: PURCHASES (/Users/iibadmin/IBM/IIBT10/workspace/MessageModellingLibrary/PURCHASES.xsd)

Name	Type	Value
PURCHASES		
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
CustomerState	xs:string	Rhode Island
CustomerCountry	xs:string	USA
CustomerMailCode	xs:string	12312
CustomerPhone	xs:string	123-123-1234
CustomerLastUpdateD	xs:string	04082008

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: Wed Mar 11 11:28:16 GMT 2015

Input

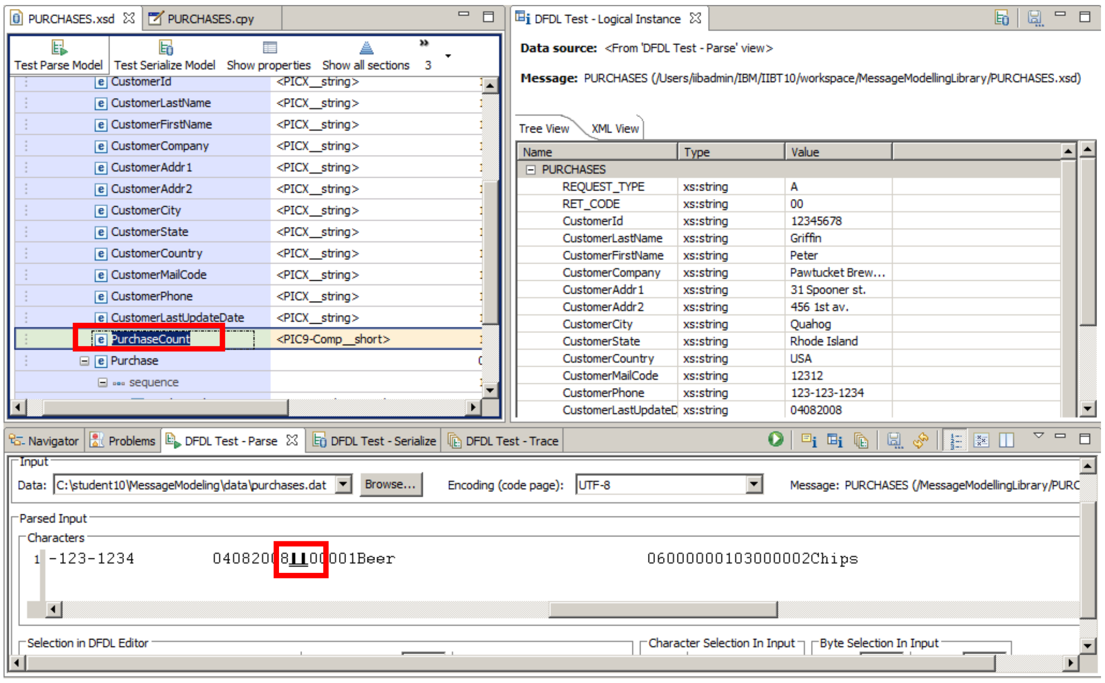
Data: C:\student10\MessageModeling\data\purchases.dat Encoding (code page): UTF-8 Message: PURCHASES (/MessageModellingLibrary/PURC

Parsed Input

Characters

```
1 A0012345678Griffin           Peter          Pawtucket Brewery           31 Spooner st.
```

10. Now click on the PurchaseCount element.



PURCHASES.xsd | PURCHASES.cpy

Test Parse Model | Test Serialize Model | Show properties | Show all sections

CustomerId	<PICX__string>
CustomerLastName	<PICX__string>
CustomerFirstName	<PICX__string>
CustomerCompany	<PICX__string>
CustomerAddr1	<PICX__string>
CustomerAddr2	<PICX__string>
CustomerCity	<PICX__string>
CustomerState	<PICX__string>
CustomerCountry	<PICX__string>
CustomerMailCode	<PICX__string>
CustomerPhone	<PICX__string>
CustomerLastUpdateDate	<PICX__string>
PurchaseCount	<PIC9-Comp__short>
Purchase	
sequence	

DFDL Test - Logical Instance

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

Message: PURCHASES (/Users/iibadmin/IBM/IIBT10/workspace/MessageModellingLibrary/PURCHASES.xsd)

Tree View | XML View

Name	Type	Value
PURCHASES		
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
CustomerState	xs:string	Rhode Island
CustomerCountry	xs:string	USA
CustomerMailCode	xs:string	12312
CustomerPhone	xs:string	123-123-1234
CustomerLastUpdateD	xs:string	04082008

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

Input

Data: C:\student10\MessageModeling\data\purchases.dat Browse... Encoding (code page): UTF-8 Message: PURCHASES (/MessageModellingLibrary/PURC

Parsed Input

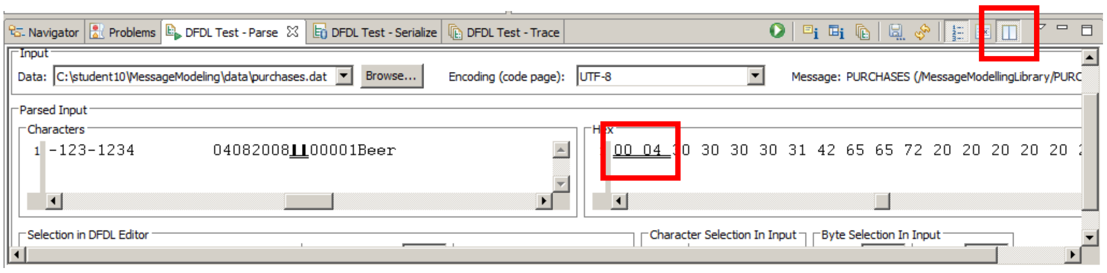
Characters

```
1 -123-1234         04082008▮▮00001Beer                     06000000103000002Chips
```

Selection in DFDL Editor | Character Selection In Input | Byte Selection In Input

Since it's a binary field, the highlighted value isn't readable with this editor.

11. Now click the "Show hex" button (top right of the lower pane, as highlighted below).



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

Input

Data: C:\student10\MessageModeling\data\purchases.dat Browse... Encoding (code page): UTF-8 Message: PURCHASES (/MessageModellingLibrary/PURC

Parsed Input

Characters

```
1 -123-1234         04082008▮▮00001Beer
```

Hex

```
00 04 30 30 30 30 31 42 65 65 72 20 20 20 20 20
```

Selection in DFDL Editor | Character Selection In Input | Byte Selection In Input

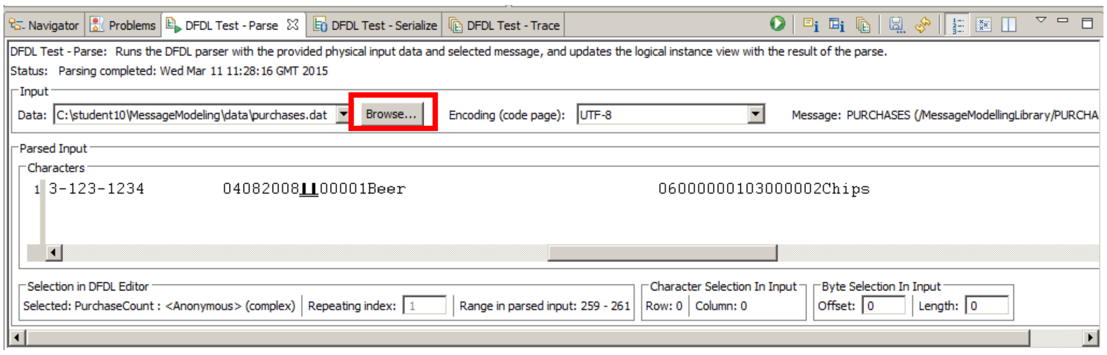
Notice that the binary field is now readable, and has a value of "00 04" which corresponds to the 4 occurrences of the "purchase" element.

Click the "Show Hex" to revert to normal display.

4. Using the Trace facility

1. Next you are going to test the message model using a malformed message.

   In the DFDL Test perspective, "DFDL Test - parse" view, click on the Browse button.



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: Wed Mar 11 11:28:16 GMT 2015

Input

Data: C:\student10\MessageModeling\data\purchases.dat

Encoding (code page): UTF-8

Message: PURCHASES (/MessageModellingLibrary/PURCHA

Parsed Input

Characters

```
1 3-123-1234         04082008  00001Beer                              06000000103000002Chips
```

Selection in DFDL Editor

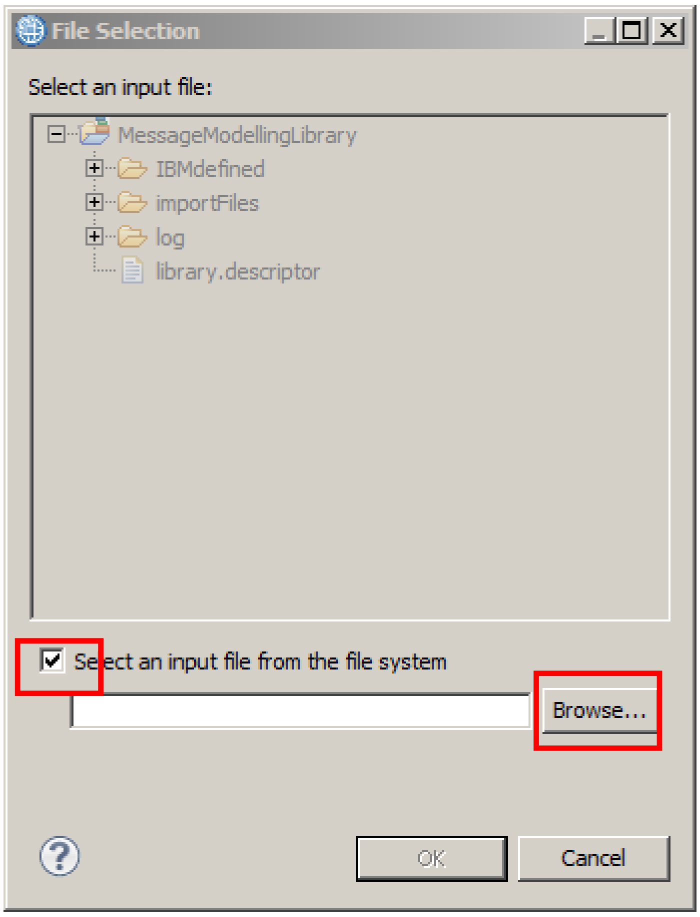
Selected: PurchaseCount : <Anonymous> (complex) Repeating index: 1 Range in parsed input: 259 - 261

Character Selection In Input: Row: 0 Column: 0

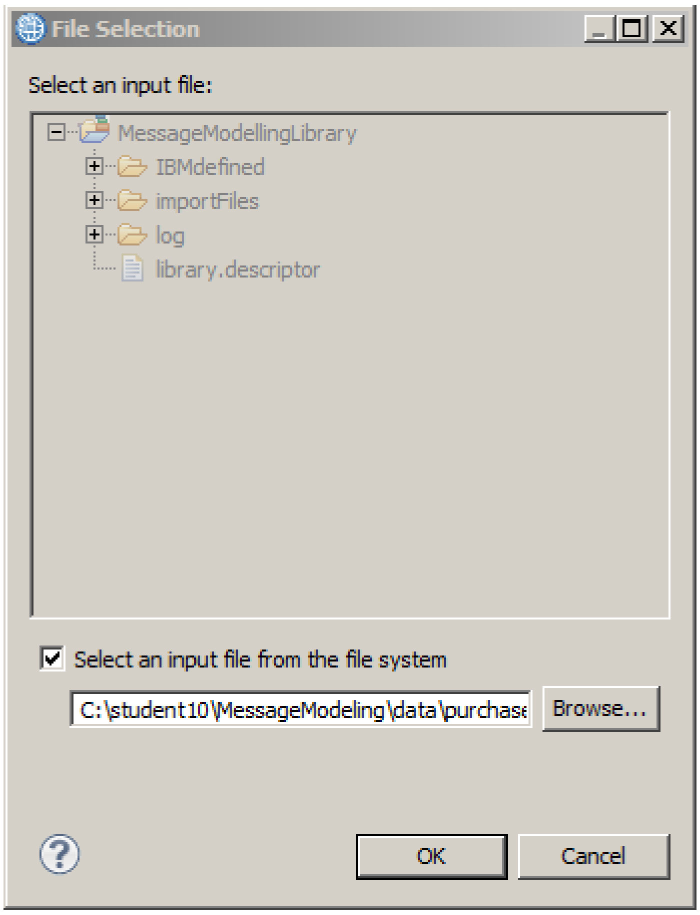
Byte Selection In Input: Offset: 0 Length: 0

2. In the File Selection dialog, select the "Select an input file from the file system" option.

   Click on the Browse button.



3. Navigate to the "C:\student10\MessageModeling\data" directory and select the "purchases_MALFORMED.dat" file.



☑ Select an input file from the file system

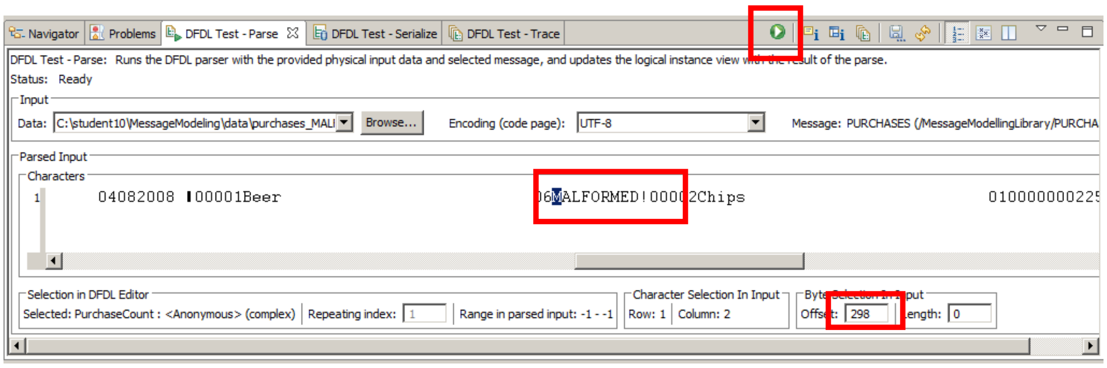
C:\student10\MessageModeling\data\purchase

Click OK.

4. In the "Offset" textbox, enter "298", and scroll right to find the highlighted character (byte 298 will be highlighted in blue).

   Note that at this position (the "Price" element position) there is a string "MALFORMED!" instead of the expected decimal number.

   Now click on the "Run parser" button to test the message model (green arrow as highlighted below).



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: Ready

Input
Data: C:\student10\MessageModeling\data\purchases_MALI Browse...
Encoding (code page): UTF-8
Message: PURCHASES (/MessageModellingLibrary/PURCHA

Parsed Input
Characters

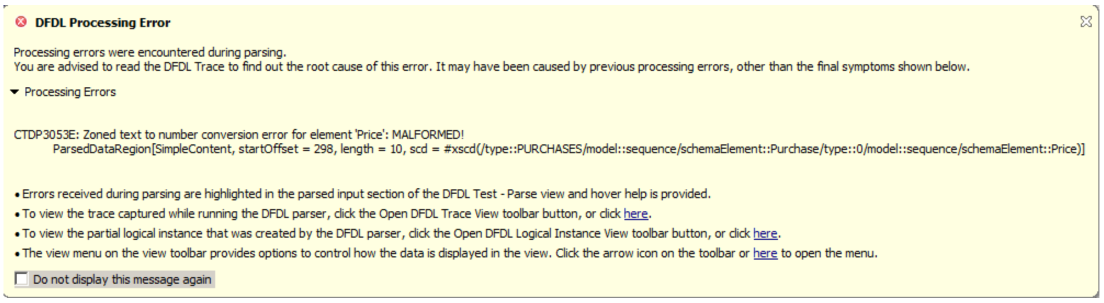
```
1       04082008 ▌00001Beer                    06MALFORMED!00002Chips                    01000000022
```

Selection in DFDL Editor
Selected: PurchaseCount : <Anonymous> (complex) Repeating index: 1 Range in parsed input: -1 - -1

Character Selection In Input
Row: 1 Column: 2

Byte Selection In Input
Offset: 298 Length: 0

5. An error message will appear with the cause of the failed parsing.



DFDL Processing Error

Processing errors were encountered during parsing.
You are advised to read the DFDL Trace to find out the root cause of this error. It may have been caused by previous processing errors, other than the final symptoms shown below.

▼ Processing Errors

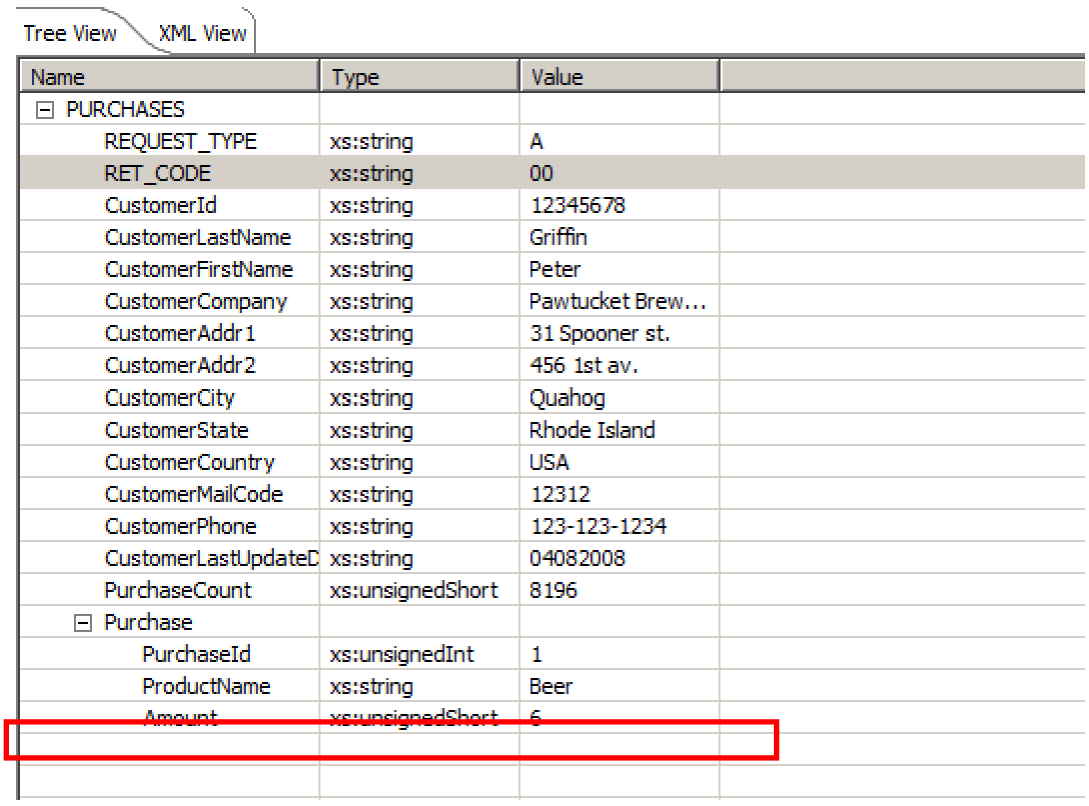
```
CTDP3053E: Zoned text to number conversion error for element 'Price': MALFORMED!
        ParsedDataRegion[SimpleContent, startOffset = 298, length = 10, scd = #xscd(/type::PURCHASES/model::sequence/schemaElement::Purchase/type::0/model::sequence/schemaElement::Price)]
```

- Errors received during parsing are highlighted in the parsed input section of the DFDL Test - Parse view and hover help is provided.
- To view the trace captured while running the DFDL parser, click the Open DFDL Trace View toolbar button, or click here.
- To view the partial logical instance that was created by the DFDL parser, click the Open DFDL Logical Instance View toolbar button, or click here.
- 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.

☐ Do not display this message again

6. Inspect the "DFDL Test - Logical Instance", you will see that the parsed tree is not complete.

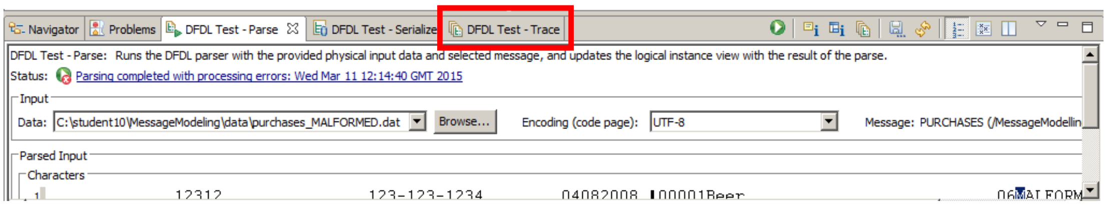
   Go to the Purchase element, expand it, and check that it was correctly parsed until the "Amount" element. The following field "Price" is empty.



Name	Type	Value
⊟ PURCHASES		
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
CustomerState	xs:string	Rhode Island
CustomerCountry	xs:string	USA
CustomerMailCode	xs:string	12312
CustomerPhone	xs:string	123-123-1234
CustomerLastUpdateD	xs:string	04082008
PurchaseCount	xs:unsignedShort	8196
⊟ Purchase		
PurchaseId	xs:unsignedInt	1
ProductName	xs:string	Beer
Amount	xs:unsignedShort	6

7. Now you will use the "DFDL Test - Trace" view, to better understand what the problem was.

   Click on the "DFDL Test - Trace" view.



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 with processing errors: Wed Mar 11 12:14:40 GMT 2015

Input

Data: C:\student10\MessageModeling\data\purchases_MALFORMED.dat Encoding (code page): UTF-8 Message: PURCHASES (/MessageModellin

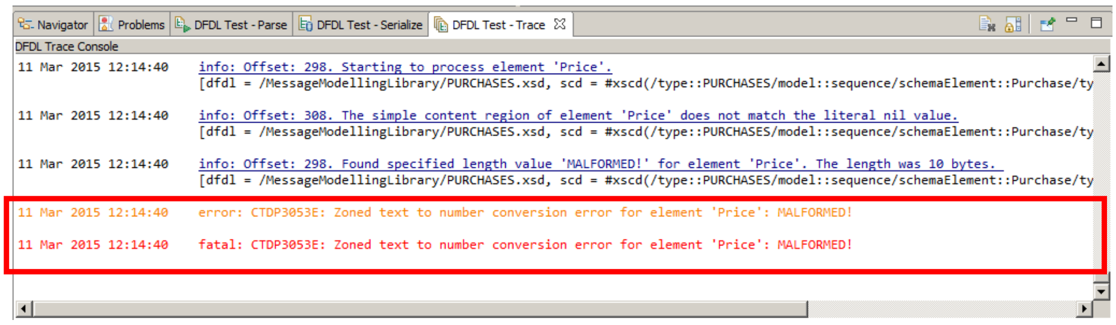
Parsed Input

Characters

```
1          12312          123-123-1234          04082008 ▌00001Beer                    06MALFORM
```

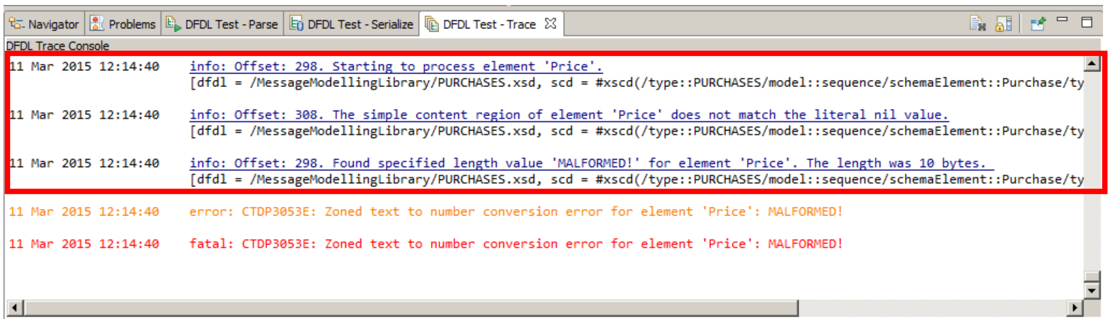
8. In the "DFDL Test - Trace" view, you will find an execution log of the parsing activities.

   At the end of the trace, there are colored lines with the found error.



```
11 Mar 2015 12:14:40    info: Offset: 298. Starting to process element 'Price'.
                        [dfdl = /MessageModellingLibrary/PURCHASES.xsd, scd = #xscd(/type::PURCHASES/model::sequence/schemaElement::Purchase/ty

11 Mar 2015 12:14:40    info: Offset: 308. The simple content region of element 'Price' does not match the literal nil value.
                        [dfdl = /MessageModellingLibrary/PURCHASES.xsd, scd = #xscd(/type::PURCHASES/model::sequence/schemaElement::Purchase/ty

11 Mar 2015 12:14:40    info: Offset: 298. Found specified length value 'MALFORMED!' for element 'Price'. The length was 10 bytes.
                        [dfdl = /MessageModellingLibrary/PURCHASES.xsd, scd = #xscd(/type::PURCHASES/model::sequence/schemaElement::Purchase/ty

11 Mar 2015 12:14:40    error: CTDP3053E: Zoned text to number conversion error for element 'Price': MALFORMED!

11 Mar 2015 12:14:40    fatal: CTDP3053E: Zoned text to number conversion error for element 'Price': MALFORMED!
```

9. Look at the lines before the error:



```
11 Mar 2015 12:14:40    info: Offset: 298. Starting to process element 'Price'.
                        [dfdl = /MessageModellingLibrary/PURCHASES.xsd, scd = #xscd(/type::PURCHASES/model::sequence/schemaElement::Purchase/ty

11 Mar 2015 12:14:40    info: Offset: 308. The simple content region of element 'Price' does not match the literal nil value.
                        [dfdl = /MessageModellingLibrary/PURCHASES.xsd, scd = #xscd(/type::PURCHASES/model::sequence/schemaElement::Purchase/ty

11 Mar 2015 12:14:40    info: Offset: 298. Found specified length value 'MALFORMED!' for element 'Price'. The length was 10 bytes.
                        [dfdl = /MessageModellingLibrary/PURCHASES.xsd, scd = #xscd(/type::PURCHASES/model::sequence/schemaElement::Purchase/ty

11 Mar 2015 12:14:40    error: CTDP3053E: Zoned text to number conversion error for element 'Price': MALFORMED!

11 Mar 2015 12:14:40    fatal: CTDP3053E: Zoned text to number conversion error for element 'Price': MALFORMED!
```

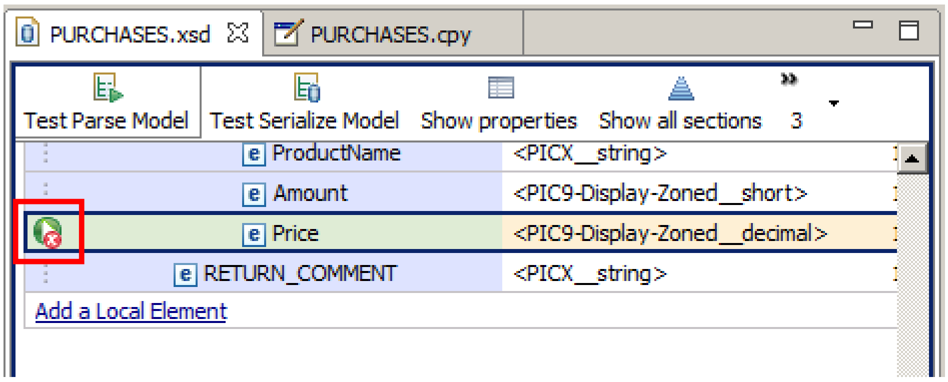
   The first line states that it is starting to process the Price element.

   In the third, it has found a string "MALFORMED!" as the value of the element.

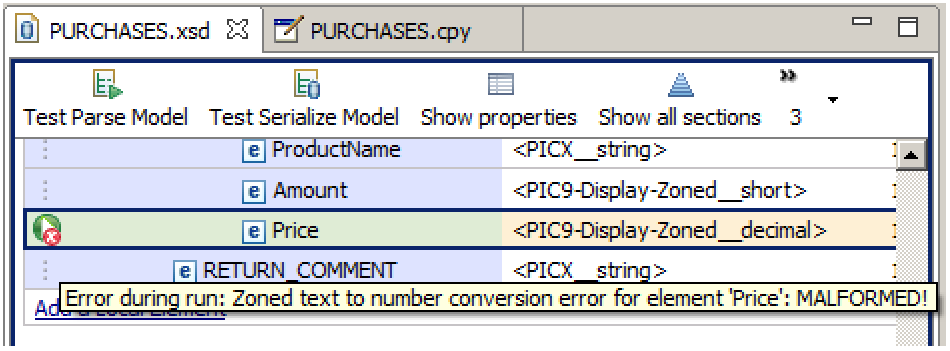
   Then the parser tries to convert the string to a decimal number, and an error appears.

10. Back in the DFDL Editor, scroll to the "Price" element.

    Note that it has an error icon next to its name.



11. Place the cursor on the error icon and a message will appear, showing the same error cause you saw in the trace view.



ProductName	<PICX__string>
Amount	<PIC9-Display-Zoned__short>
Price	<PIC9-Display-Zoned__decimal>
RETURN_COMMENT	<PICX__string>

```
Error during run: Zoned text to number conversion error for element 'Price': MALFORMED!
```

END OF LAB GUIDE