16L94

IBM Integration Bus

Message Modeling with DFDL

Lab 4
Record-oriented, tagged, delimited text (advanced)

Featuring:

- Schema references
- Initiators, separators, terminators
- Arrays - element count provided explicitly within data
- Unordered sequences
- Arrays - element count given by parsing data

June 2015

Hands-on lab built at product Version 10.0.0.0

1. INTRODUCTION 3

1.1 LAB OBJECTIVES 3
1.2 LAB PREPARATION 3
1.3 TDS BASIC LAB RECAP 4
1.4 TDS ADVANCED SCENARIO 5

2. CREATE THE MESSAGE MODEL IN THE LIBRARY 6

3. REFINE THE MESSAGE MODEL 12

4. TESTING THE MESSAGE MODEL 31

5. ESTABLISHING REPEATING ELEMENTS BY PARSING 37

6. UNORDERED SEQUENCES 42

END OF LAB GUIDE 45

Provided by IBM BetaWorks

1. Introduction

1.1 Lab objectives

In this lab you are going to extend the message model you created in the previous (basic TDS) lab, Lab3.
The lab examines more complex structures, such as schema references, element delimiters like initiators, separators and terminators, and the ability to handle variable numbers of elements.

Chapters 5 and 6 contains scenarios that are supported only in IIB V9.0.0.2 and later (inc V10). These scenarios are arrays where the count does not occur in the data and is established solely by parsing (by setting the Occurs Count Kind property to "parsed"), and sequences where the elements occur in any order (by setting the Sequence Kind property to "unordered").

1.2 Lab preparation

If you did not do Lab 3 (the basic TDS lab), you can import a pre-built solution. Import the PI file

c:\student10\message_modelling\solution
MessageModelSolution_TDS_Basic_Lab3(SharedLib).zip

If you wish to proceed straight to the new function provided in IIB V9002 (Occurs Count Kind=parsed in chapter 5 and unordered sequences in chapter 6), import the partial solution of this lab:

c:\student10\message_modelling\solution
MessageModelSolution_TDS_Advanced_Lab4(SharedLib-PartialSolution).zip

A complete solution of this lab is provided in:

c:\student10\message_modelling\solution
MessageModelSolution_TDS_Advanced_Lab4(SharedLib-FullSolution).zip

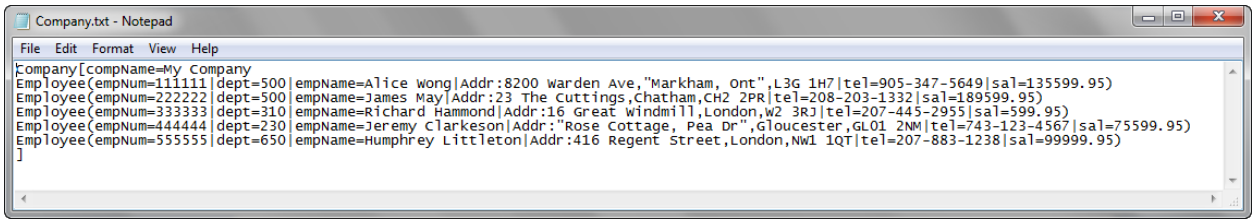
If you import one of the PI files provided above, the imported library will be called

MessageModelling_TDS

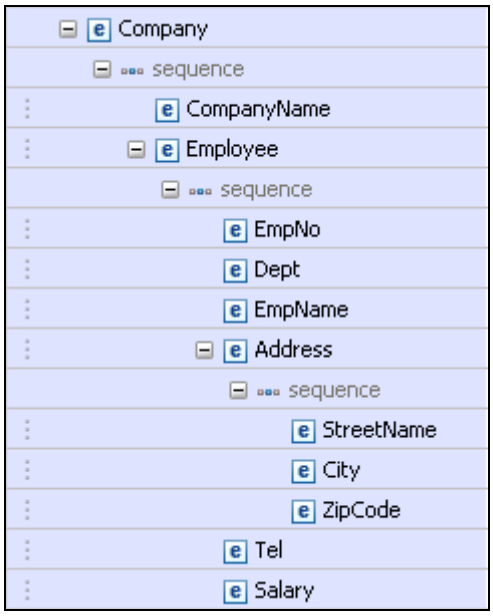
Provided by IBM BetaWorks

1.3 TDS Basic Lab Recap

In the TDS basic lab, you created a message model to parse this data file. The file has a single Company record which contains multiple Employee records.



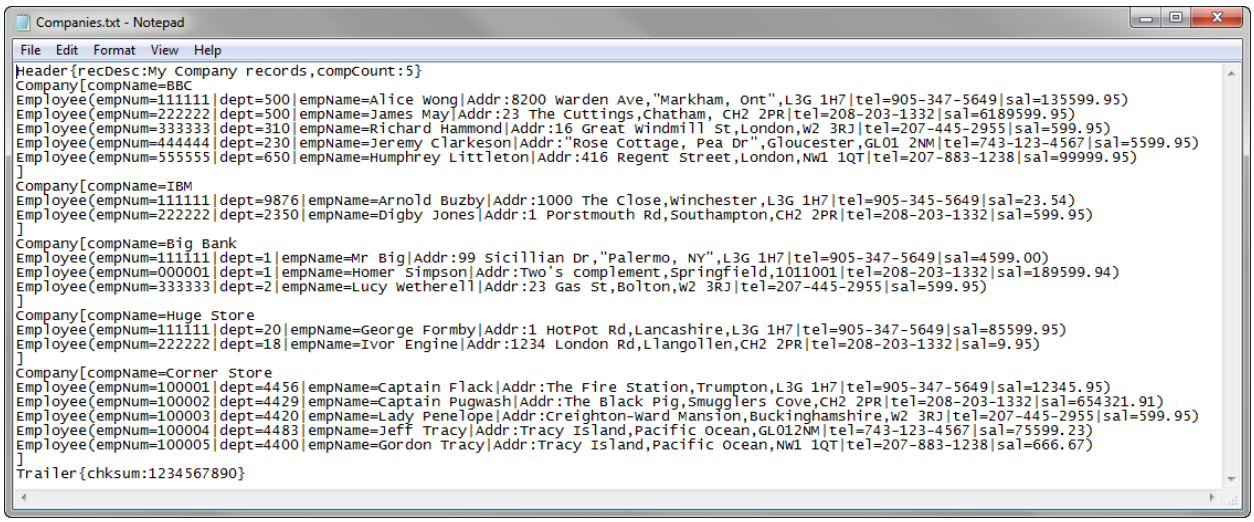
So you defined the following DFDL structure like this:



1.4 TDS Advanced Scenario

In this lab, you will extend the message model created in the Basic scenario so that it can parse the following data file, which contains multiple Company records.

In the first part of this lab (chapter 3), the number of Company records is specified by the compCount element in the Header record (the supplied test file has a value of 5).



So you will need to create a structure like this. The Companies schema will use a schema reference to the Company schema that was generated in Lab3.

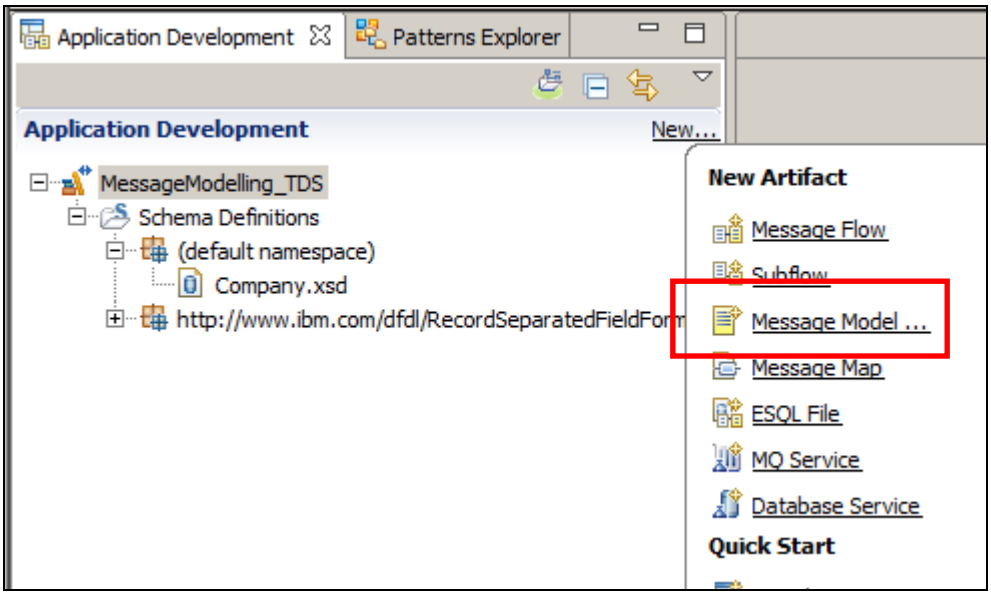
In the next part, chapter 5 shows you how to automatically detect the number of company records. The DFDL parser can be defined to detect the number of Company elements without using the compCount element.

Finally, chapter 6 shows you how to switch the order of the Header elements without having to redefine the order of the elements in the model.

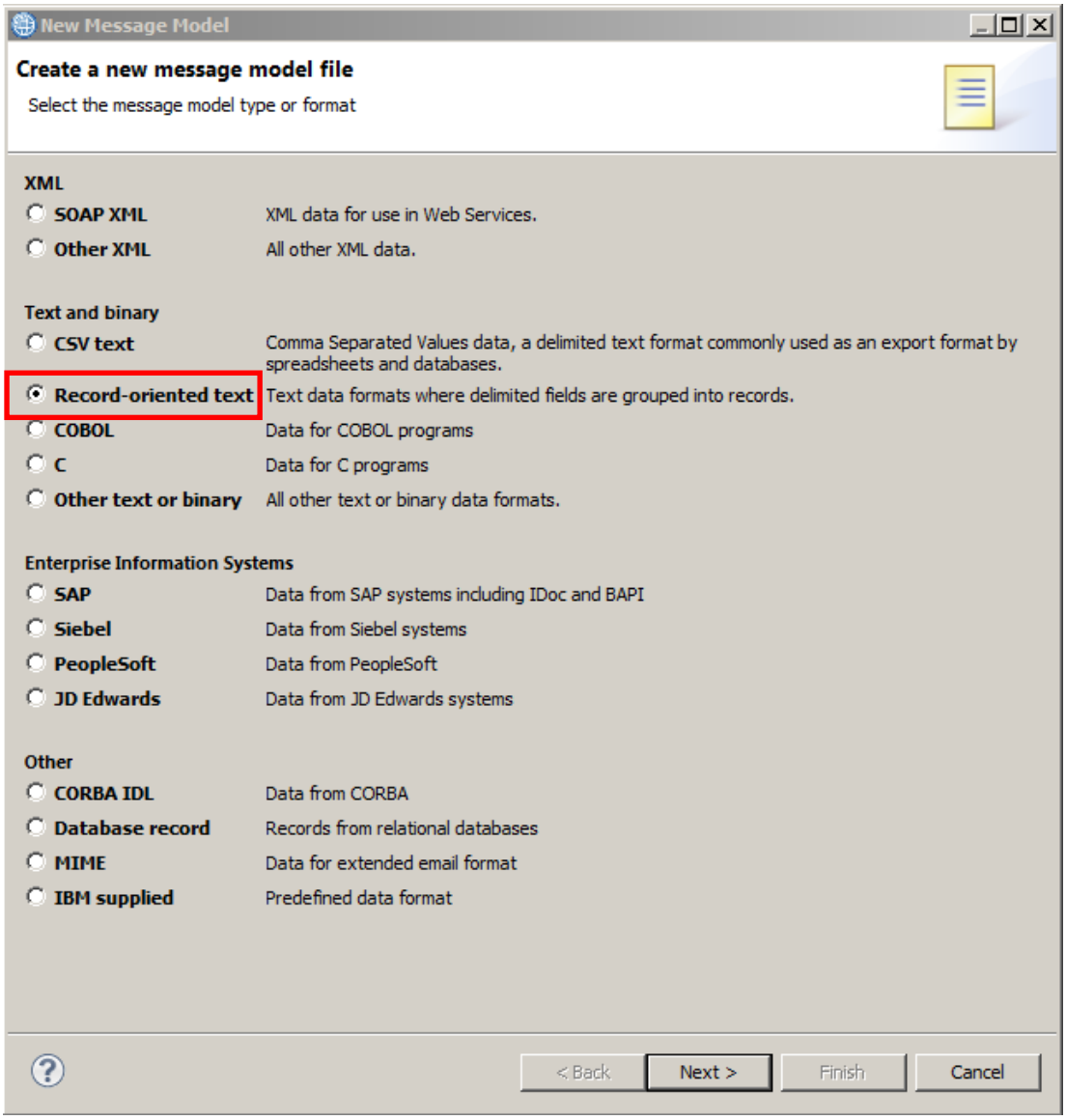
2. Create the Message Model in the Library

1. Use either the library you created for Lab 3 (MessageModelingLibrary), or the provided solution for Lab 3 (MessageModelling_TDS).

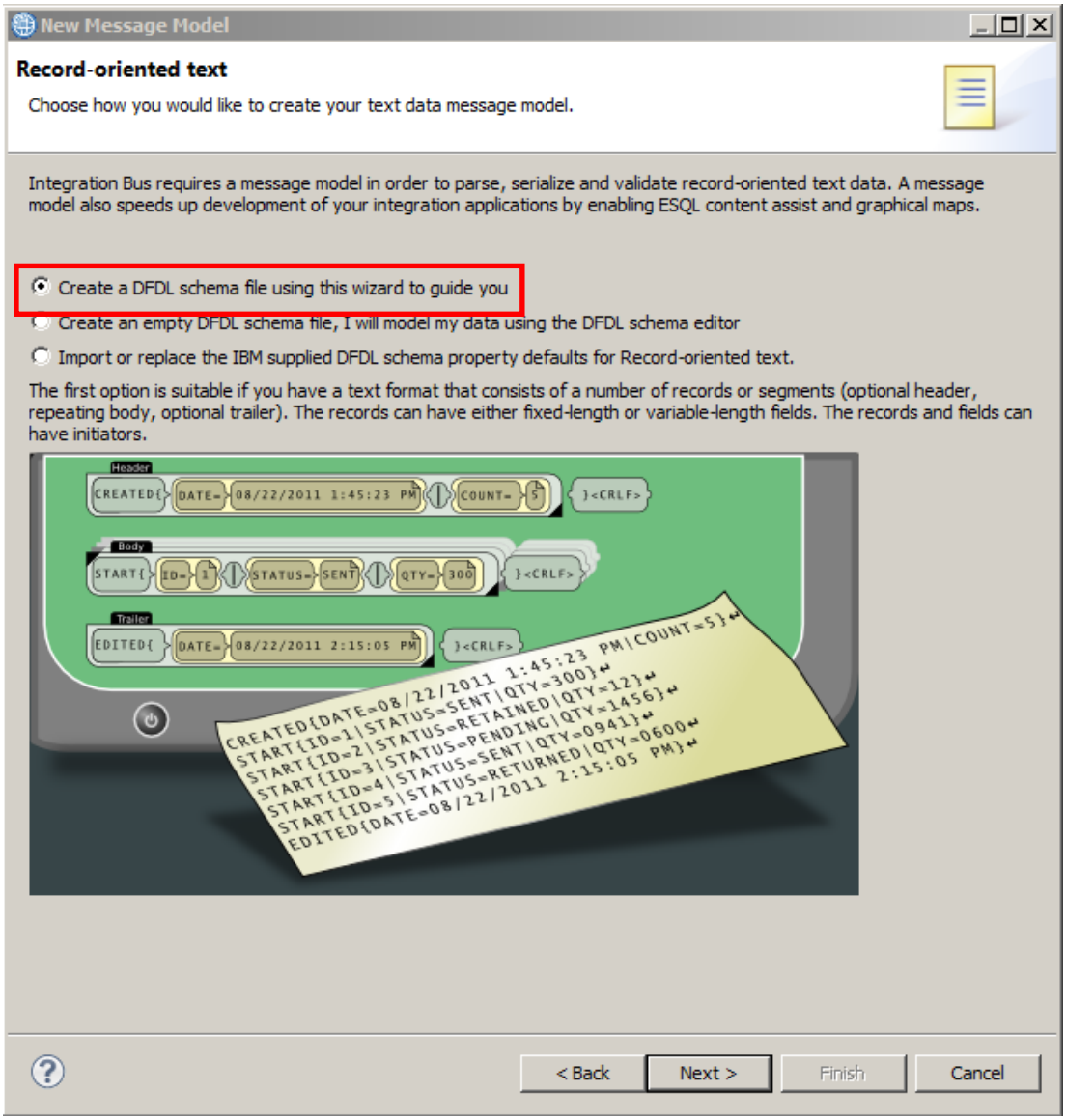
   Click New->Message Model to create a new DFDL Schema.



2. In the "New Message Model" wizard, select "Record-oriented text" and click Next.

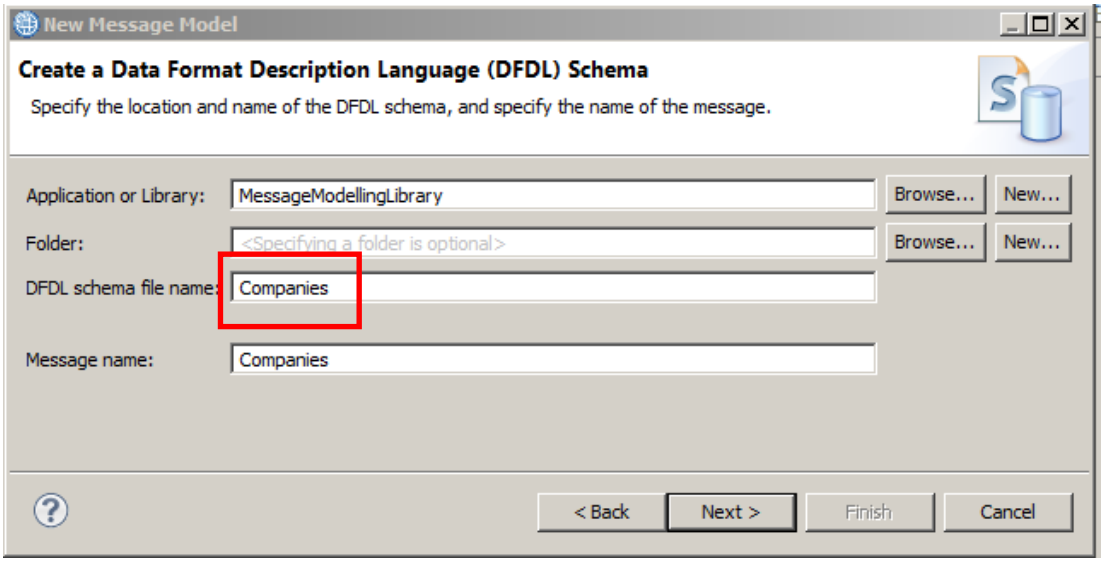


3. From the wizard, select "Create a DFDL schema file using the wizard to guide you" and click Next.



4. Enter "Companies" as the DFDL Schema file name and ensure the Library name is set to the correct value (our example is using MessageModelling_TDS).

   Click Next.



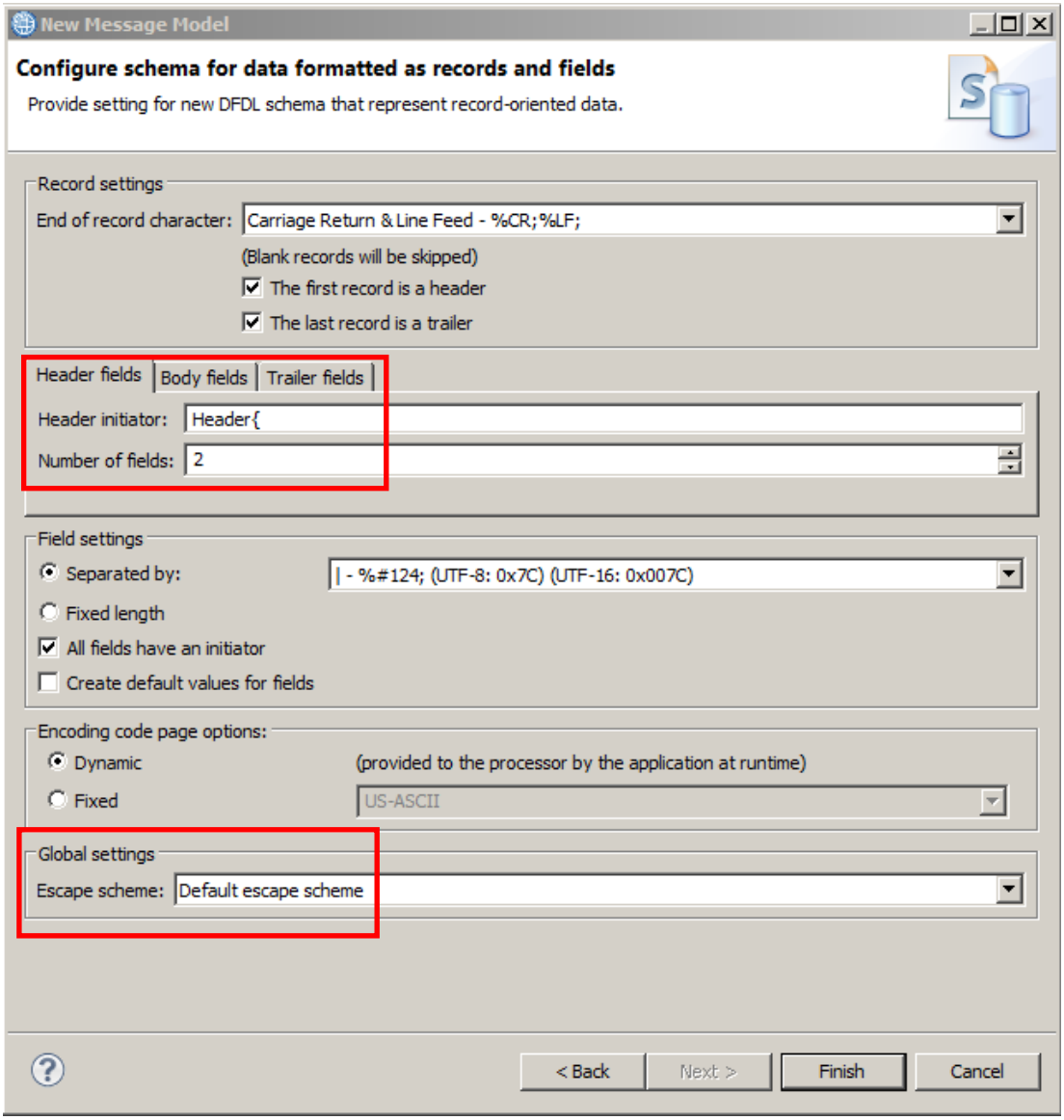
Note that the Message name will auto complete based on the DFDL schema file name.

5. Leave the "End of record character" default value (carriage return, line feed).

   Leave the "The first record is a header" and "The last record is a trailer" checked.

   In the Header fields tab, enter "Header{" as the Header initiator, and set "Number of fields" = 2.

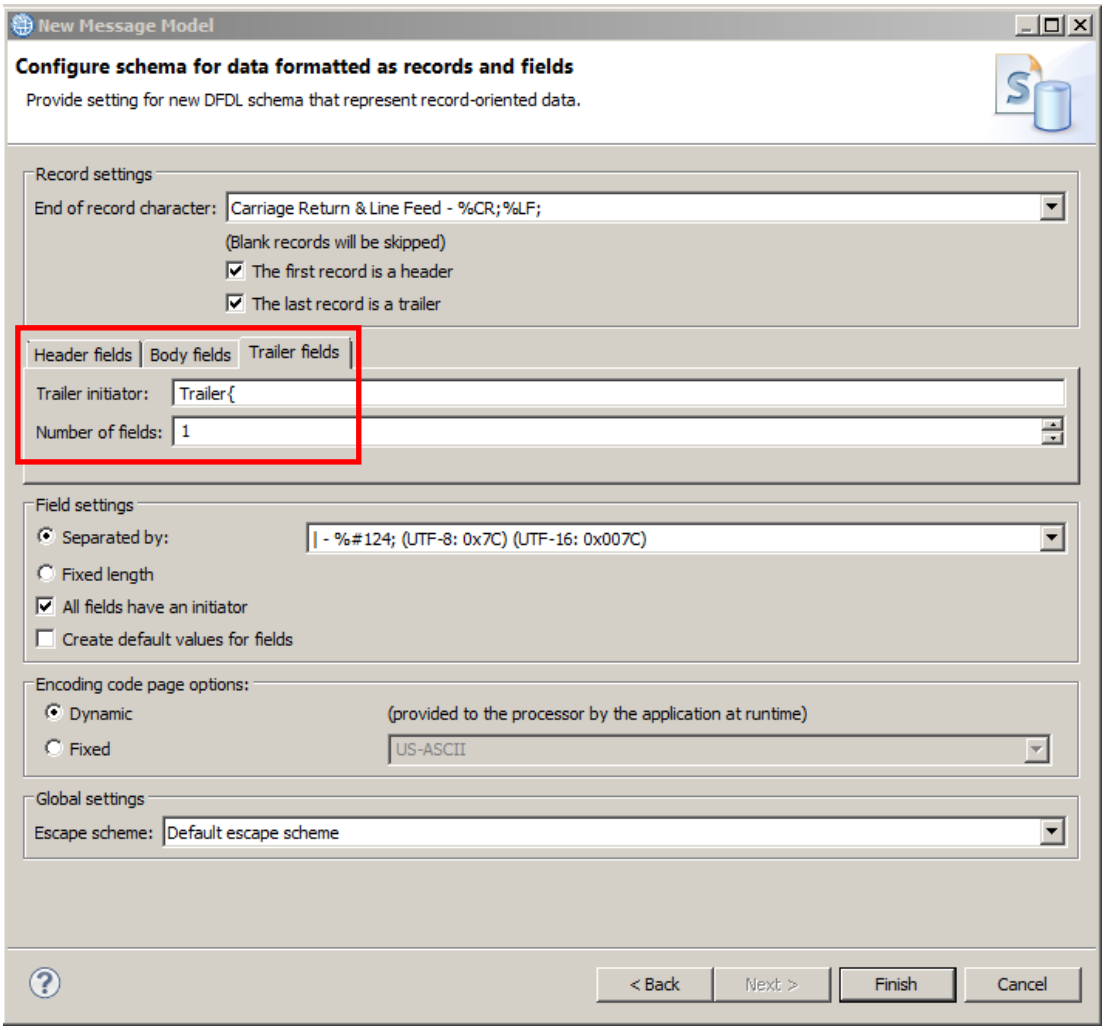
   Change the Escape scheme to "**Default escape scheme**". Note that in versions of IIB prior to V9.0.0.2, the Escape scheme was automatically set to this value. The default escape scheme is required in this lab, because there is an element in the test data which has a value containing embedded comma (,) which needs to be escaped.



Provided by IBM BetaWorks

6. Click on the "Trailer fields" tab, and enter "Trailer{" as the Trailer initiator and set "Number of fields" = 1.

   Click Finish.



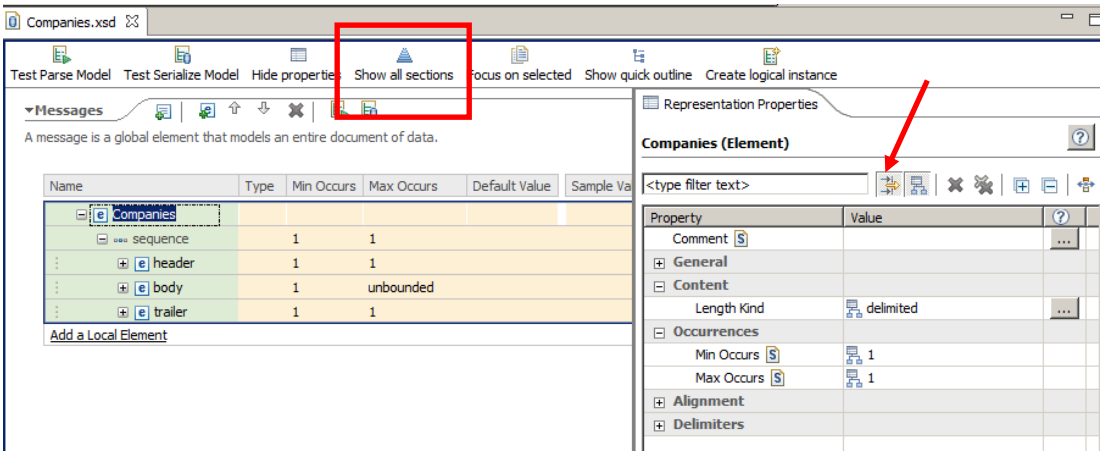
Provided by IBM BetaWorks

3. Refine the Message Model

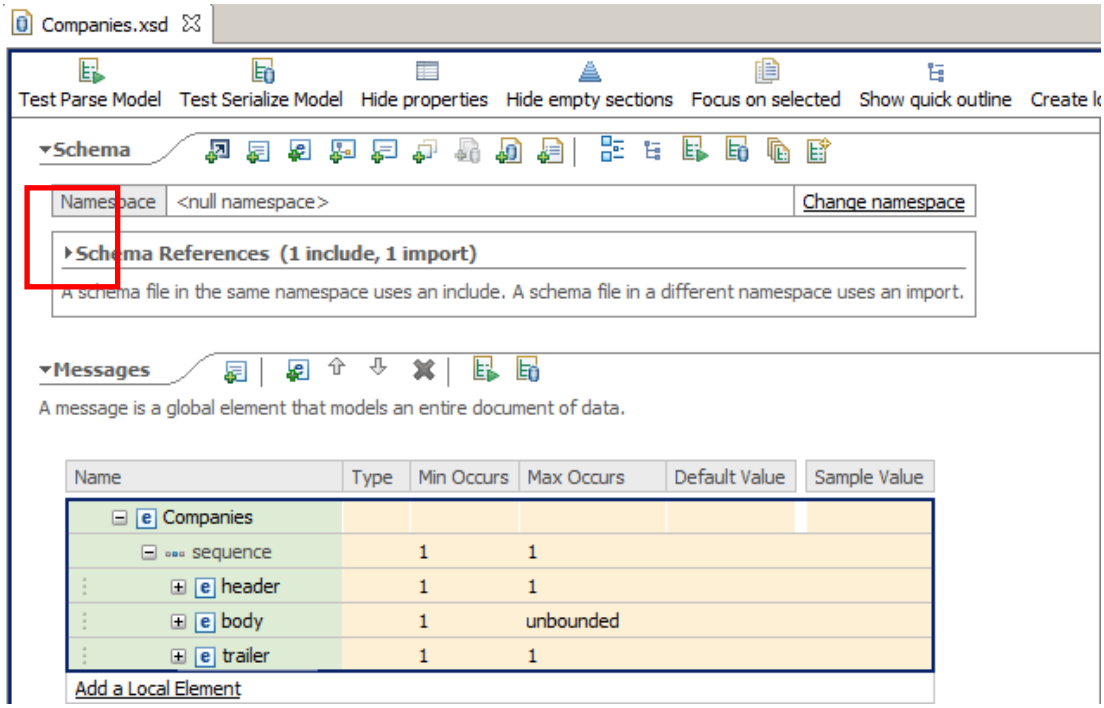
1. The DFDL editor will open with the generated DFDL Schema.

   Click the "Show all sections" icon.

   If you are using IIB V9.0.0.1 or earlier, you will also see the "Show Advanced" icon. You should click this icon now. In V9.0.0.2 (and in V10), this icon has been removed, and has been replaced by a small icon in the Representation Properties (see the arrowed icon below). The default is that Advanced properties are shown.

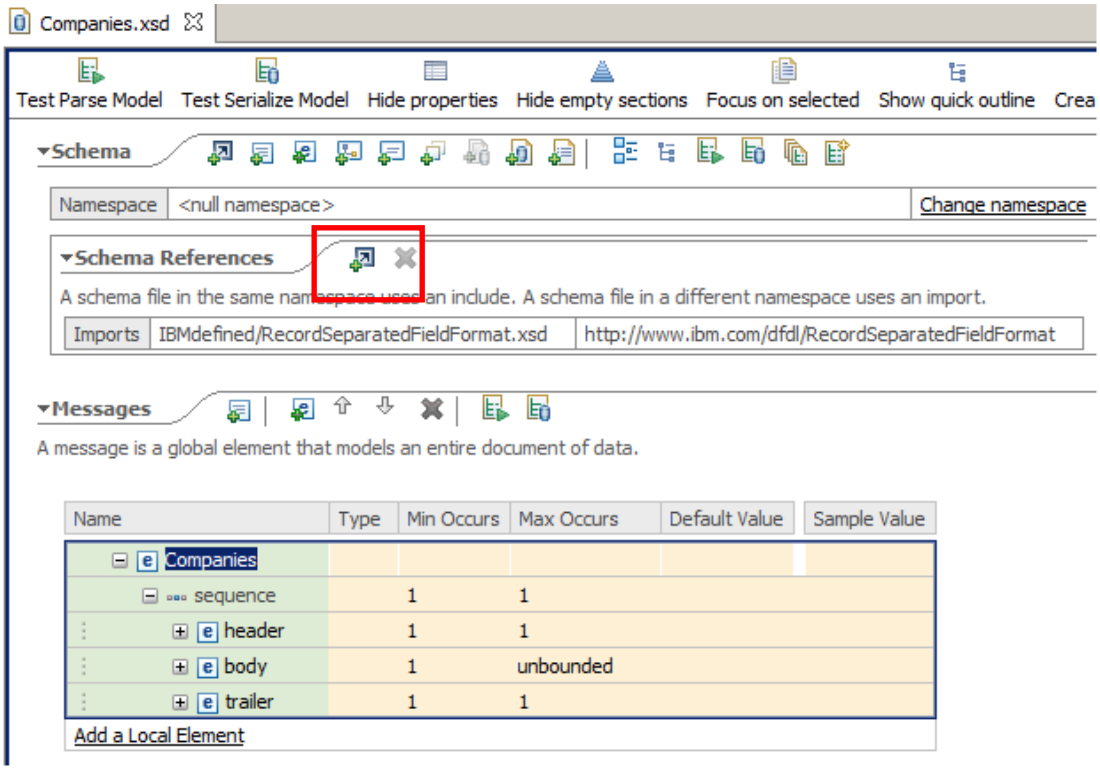


2. Click the twisty next to the "Schema References" section to expand the references (Includes and Imports) of the DFDL Schema file.

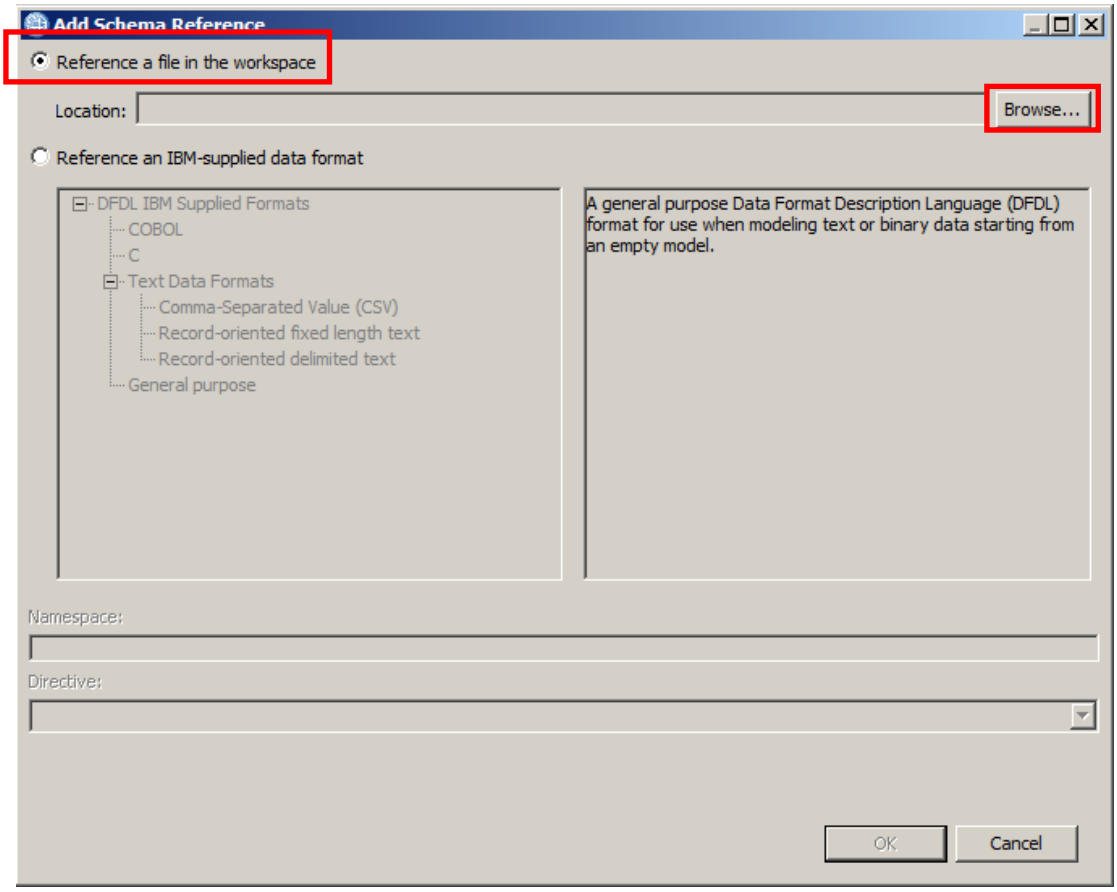


3. This message model is going to build on the message model from the basic TDS lab, by creating a reference to the Company schema.

   Click the "Add a reference to another schema" icon.

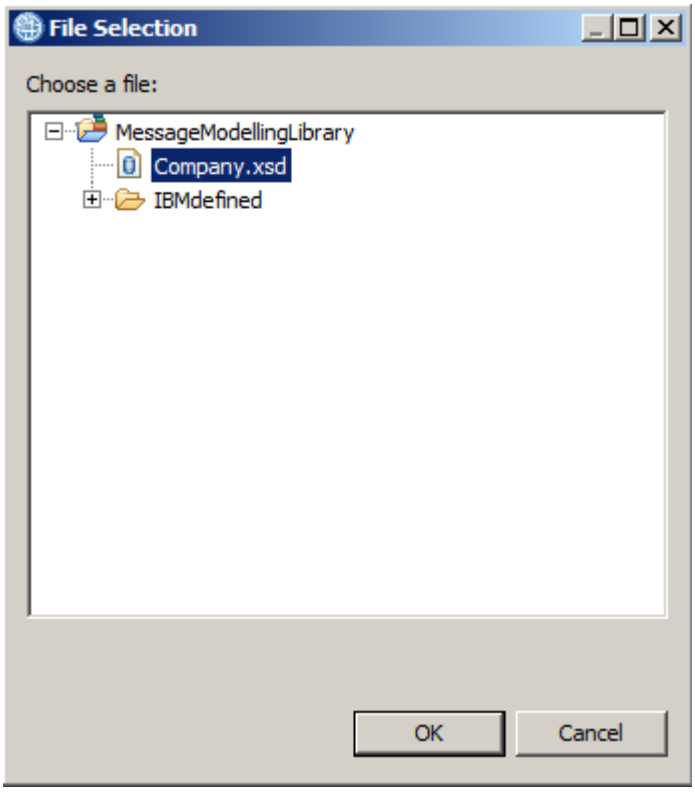


4. In the "Add Schema Reference" window, leave the default "Reference a file in the workspace" option and click Browse.



5. Select the "Company.xsd" file you created in the previous (basic) lab, or the one in the pre-built library, MessageModelling_TDS.

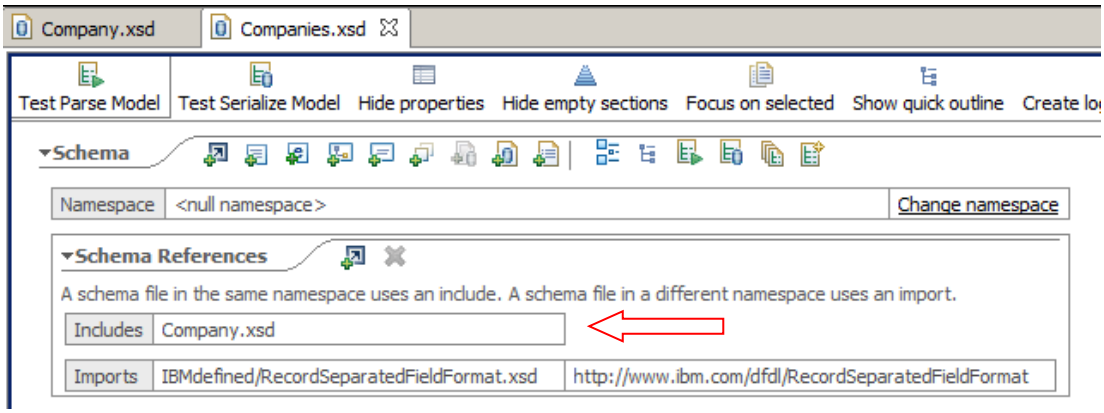
   Click OK.



   Back in the "Add Schema Reference" window click OK.

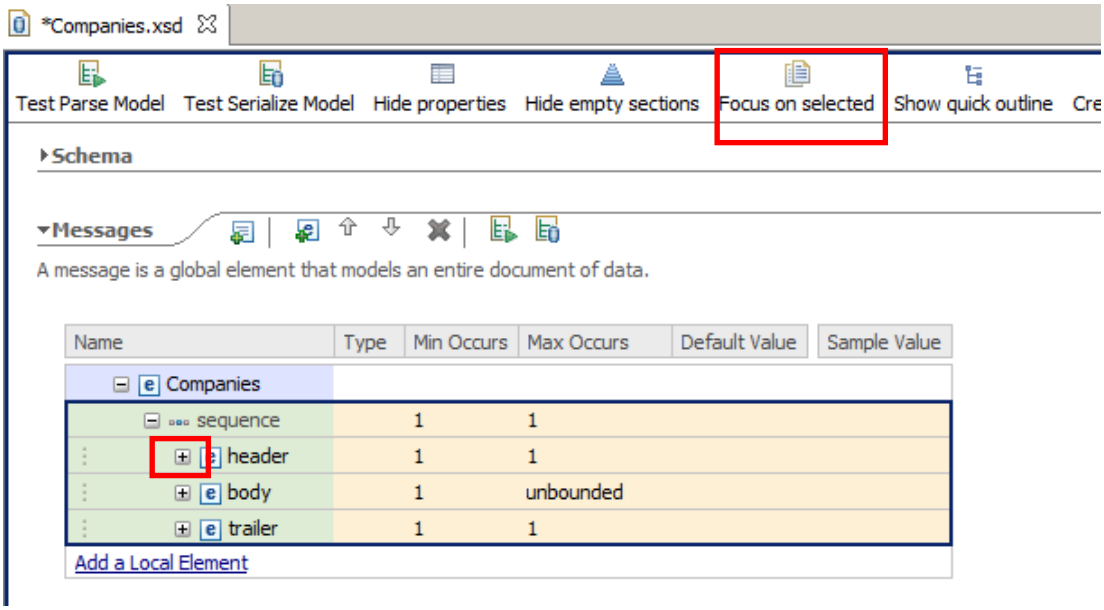
6. Note that you have two Schema References:

   - Company.xsd (which you've just added).
   - RecordSeparatedFieldFormat.xsd (which was automatically added by the wizard): This contains Record Separated specific defaults for DFDL properties. This schema already existed in the library (it is an IBM-defined schema), and Companies.xsd now has a reference to this existing schema**.**

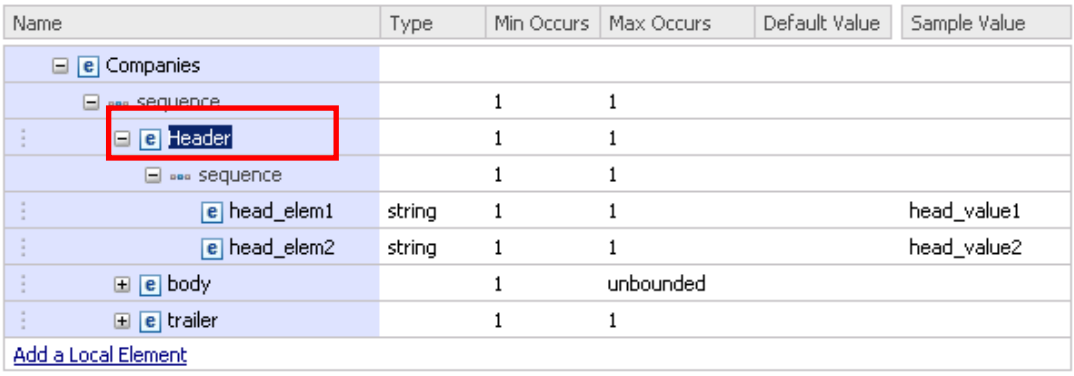


7. Click anywhere inside the Companies Message section, and click on the "Focus on selected" icon.

   Then, expand the header element by clicking the "+".

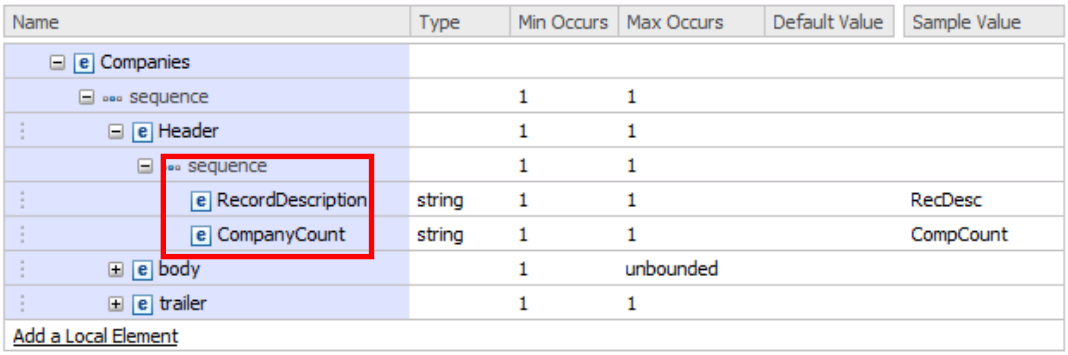


8. Change the header element's name to "Header" (capital "H") by highlighting it, and overtyping.

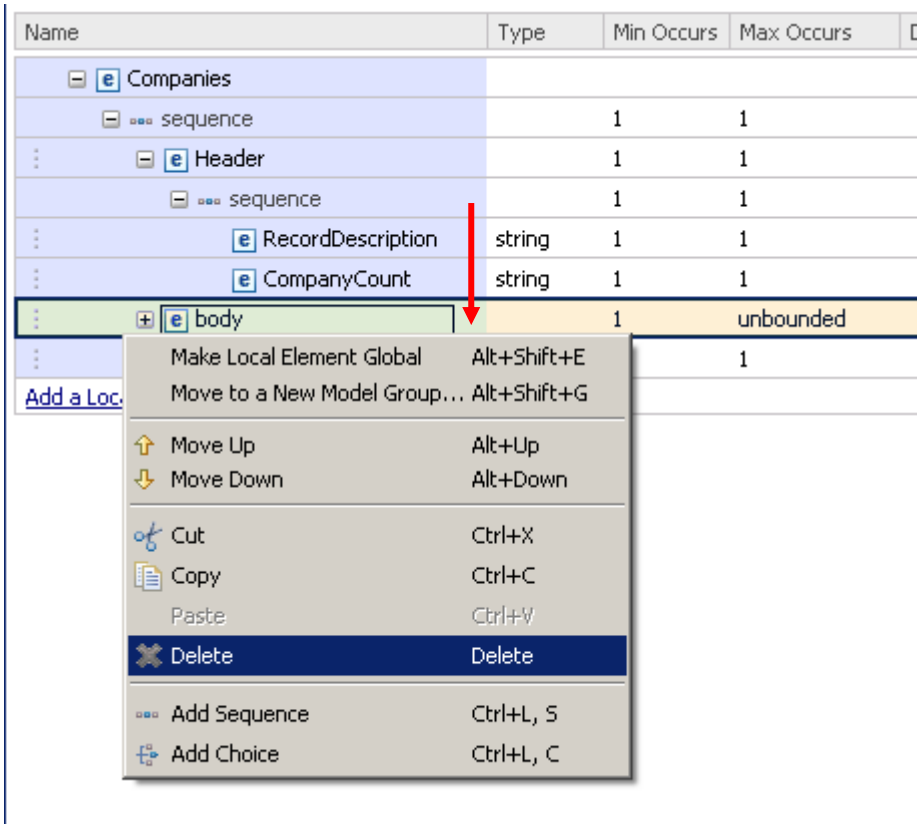


9. Change the 2 elements under Header to "RecordDescription" and "CompanyCount".

   If you wish, you can also change the sample values, as shown.

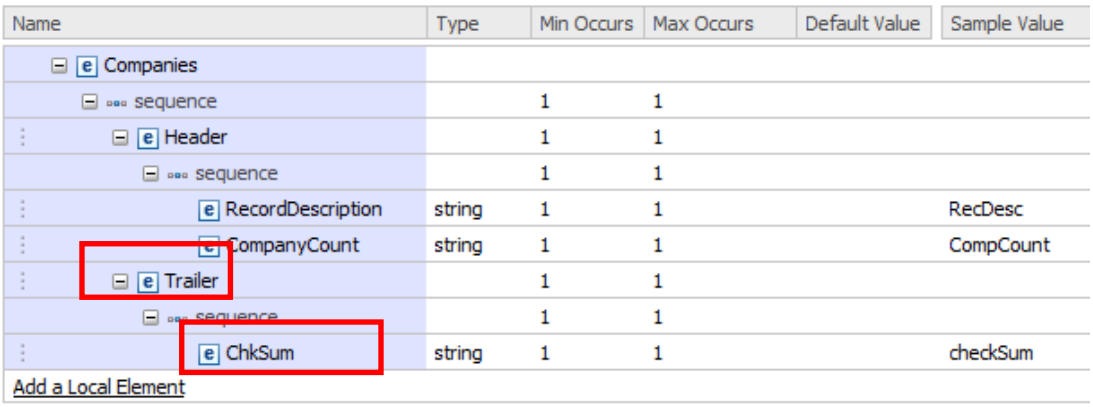


10. Delete the "body" element by right-clicking on the line of the element and selecting Delete. (Do not right-click on the text of the element name … you will see a different context menu).



11. Change the trailer element's name to "Trailer" (capital "T").

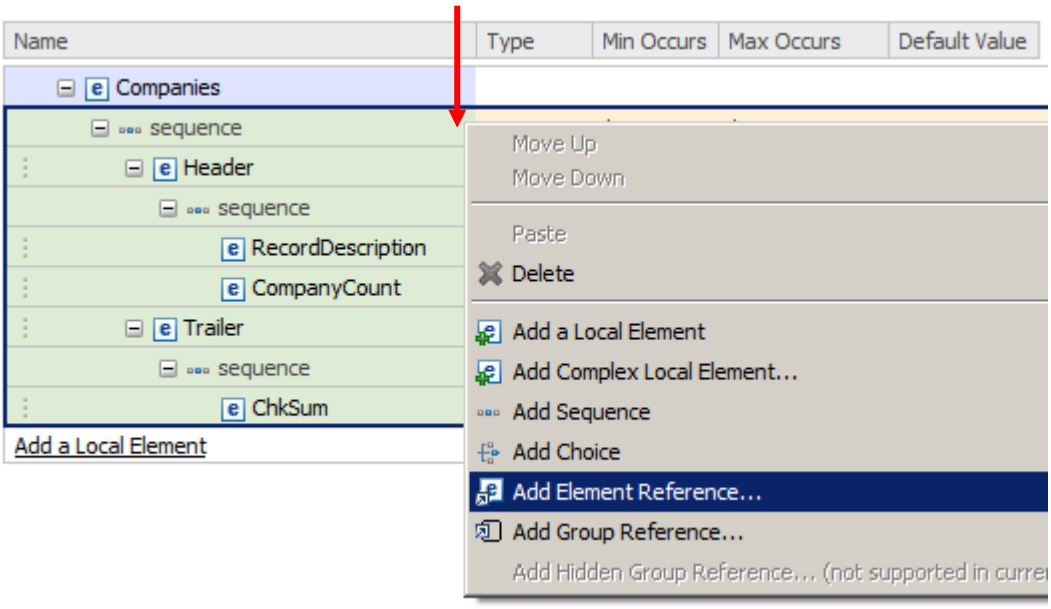
    Change the name of the element under Trailer to "ChkSum".



12. Earlier, you added a reference to the Company.xsd schema.

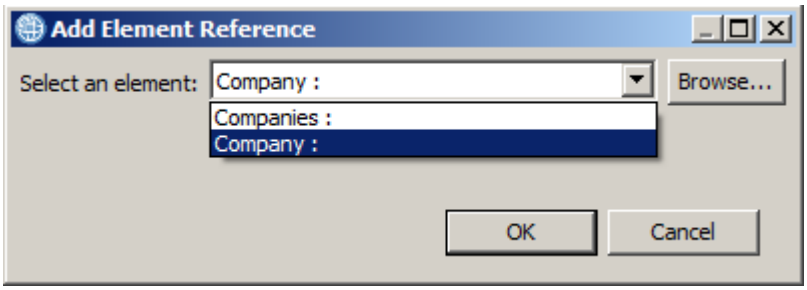
    You are now going to use this reference, by adding an Element Reference in the appropriate part of the Companies message.

    To do this, right-click on the Companies' "sequence" element and select "Add Element Reference". Note - you must position the mouse to the right of the word "sequence". If you are too close, a different context menu will appear.



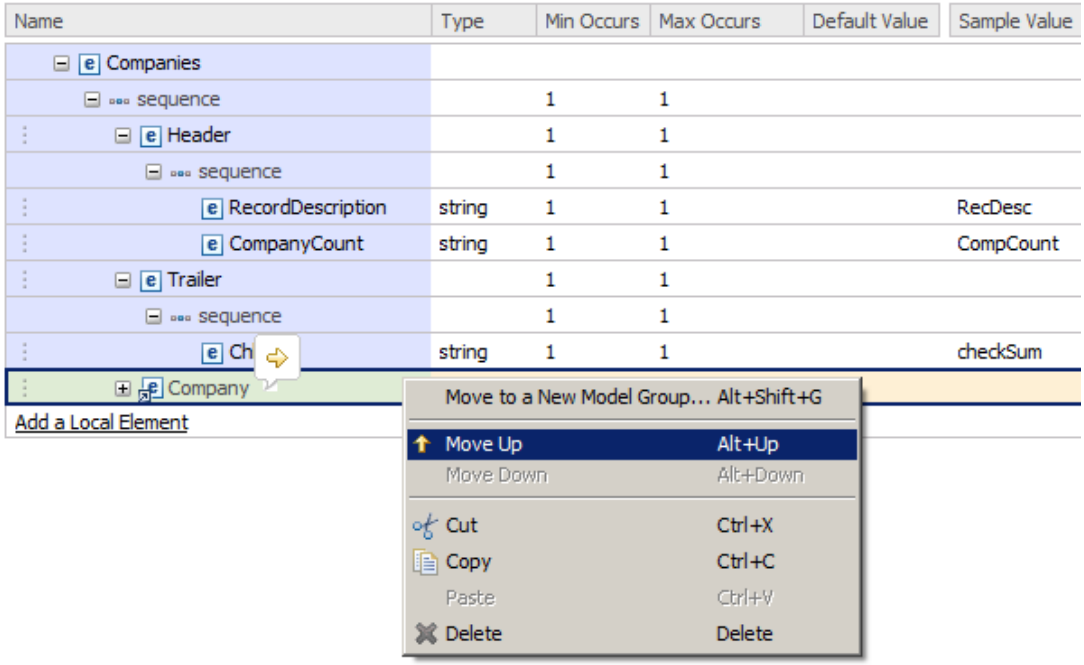
13. Select "Company :" from the dropdown

    Click OK.



Provided by IBM BetaWorks

14. The new element has to be positioned correctly, so right-click on the newly added element reference, and select "Move Up".



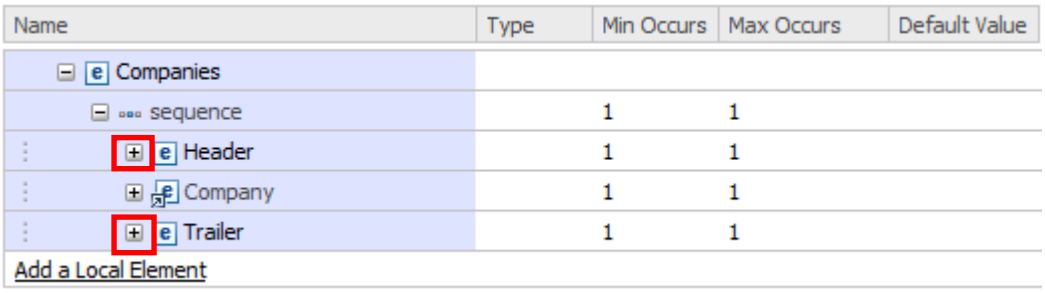
Note that the added element reference has a different icon to differentiate from a regular element (this is only a reference to an existing element in another DFDL schema).

Also note that the element reference's name is greyed out because it is read-only. To modify it you will need to open the DFDL schema where it was defined clicking on the yellow arrow that appears when you hover over the element.( ).

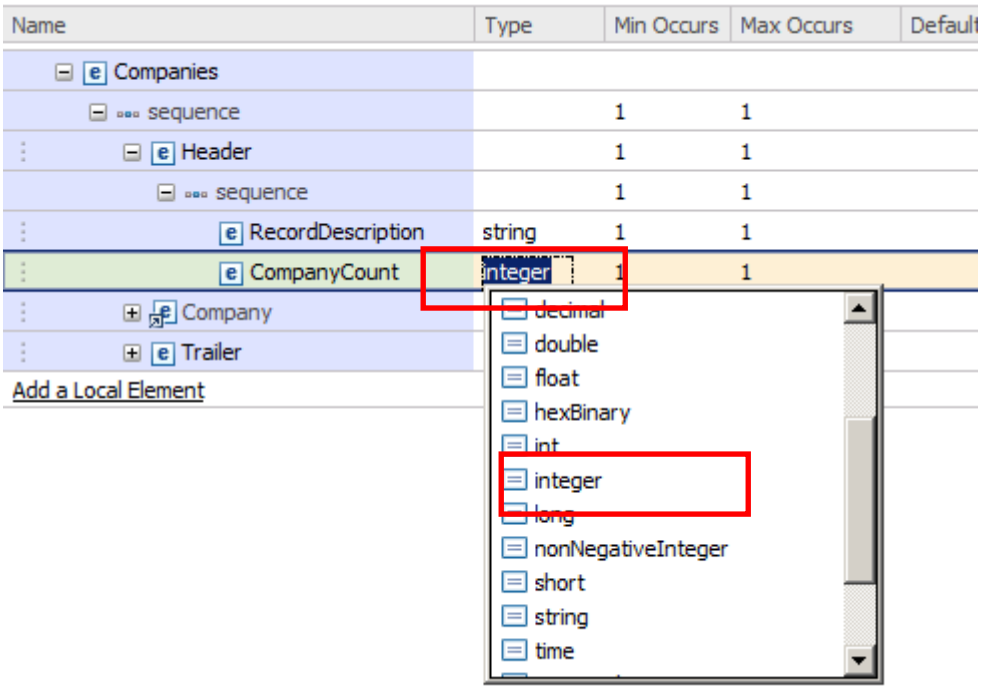
When you have moved the Company element, the model will look like this.

Name	Type	Min Occurs	Max Occurs	Default Value	Sample Value
Companies					
sequence		1	1		
Header		1	1		
sequence		1	1		
RecordDescription	string	1	1		RecDesc
CompanyCount	string	1	1		CompCount
Company		1	1		
sequence		1	1		
CompanyName	string	1	1		*a*
Employee		1	unbounded		
Trailer		1	1		
sequence		1	1		
ChkSum	string	1	1		checkSum
Add a Local Element					

15. Expand both the Header and Trailer elements by clicking the "+" next to them.

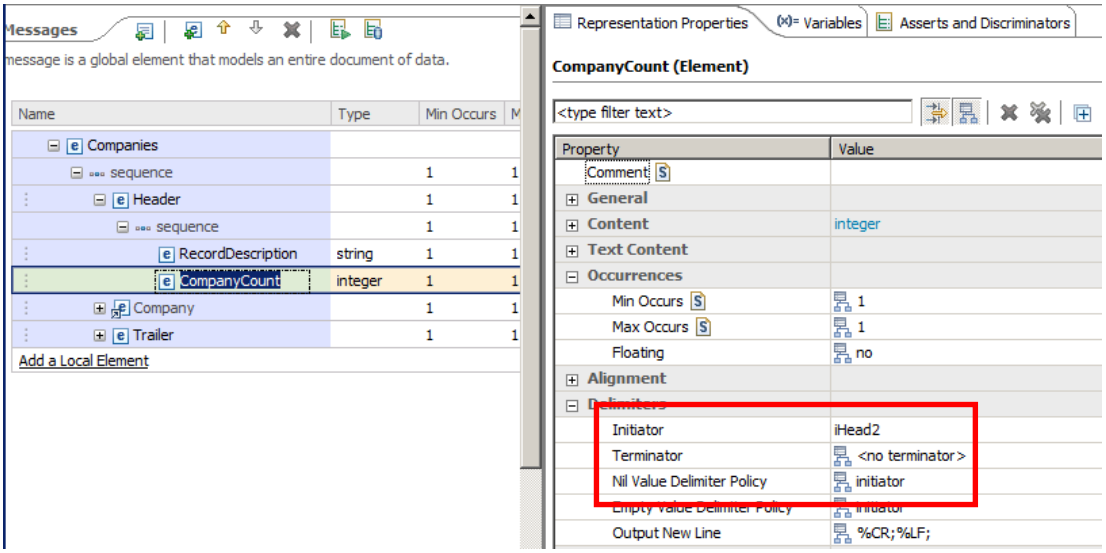


16. Change the CompanyCount element type to "integer" by clicking its type column and selecting "integer" (not int).

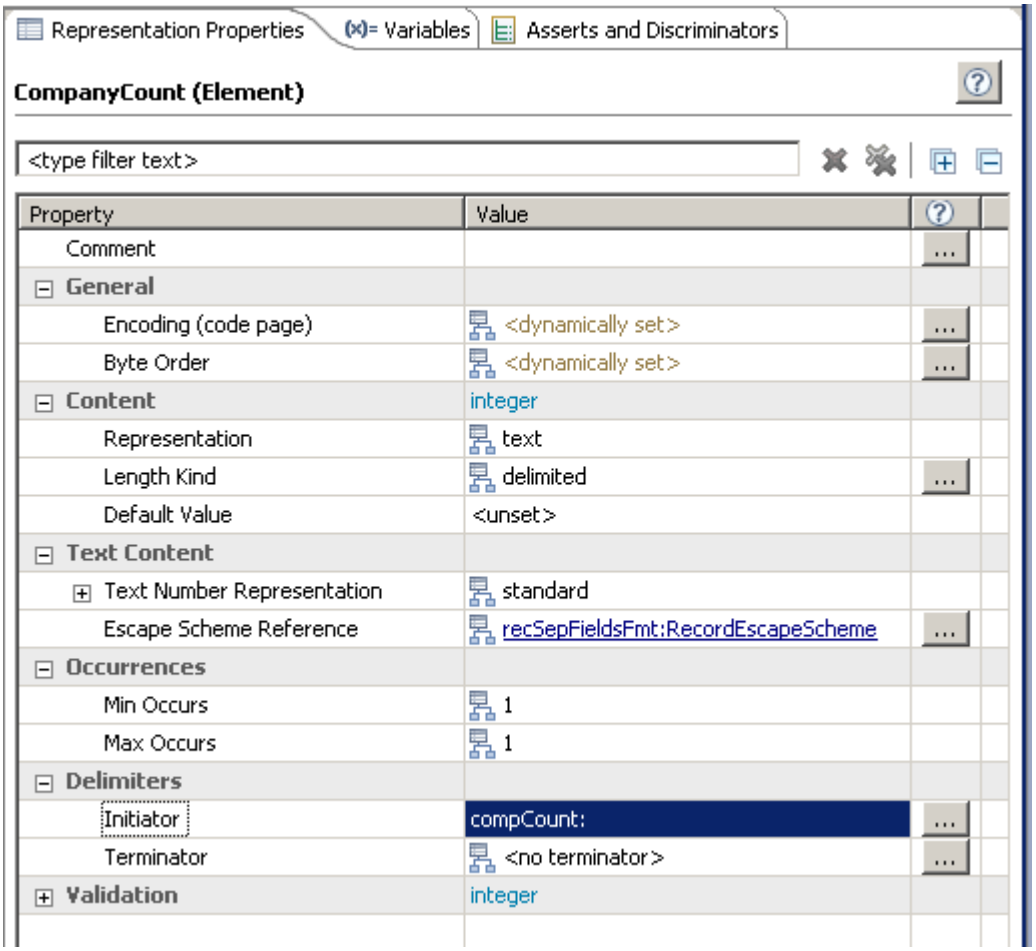


Provided by IBM BetaWorks

17. Click the CompanyCount element again, and look at the "Delimiters" section in the Representation Properties.



18. Change the Initiator's value ("iHead2") to "compCount:" **(don't miss out the : (colon)).**



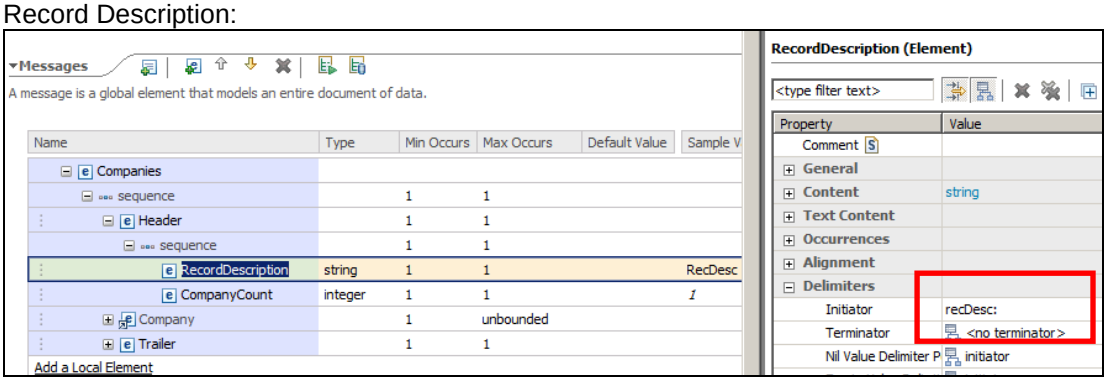
Provided by IBM BetaWorks

19. Repeat the previous step for the "RecordDescription" and ChkSum fields with the following values:

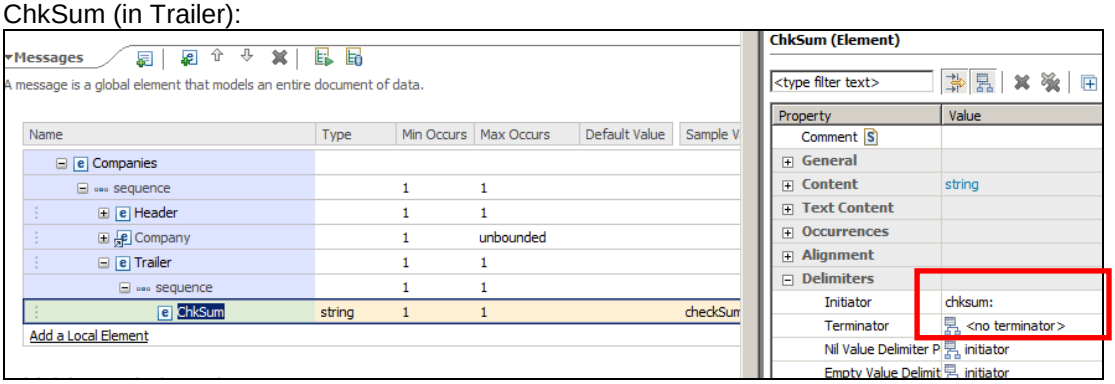
RecordDescription	recDesc:
ChkSum	chksum:

Again, do not miss the colon characters. The case of the property values is important.

Record Description:

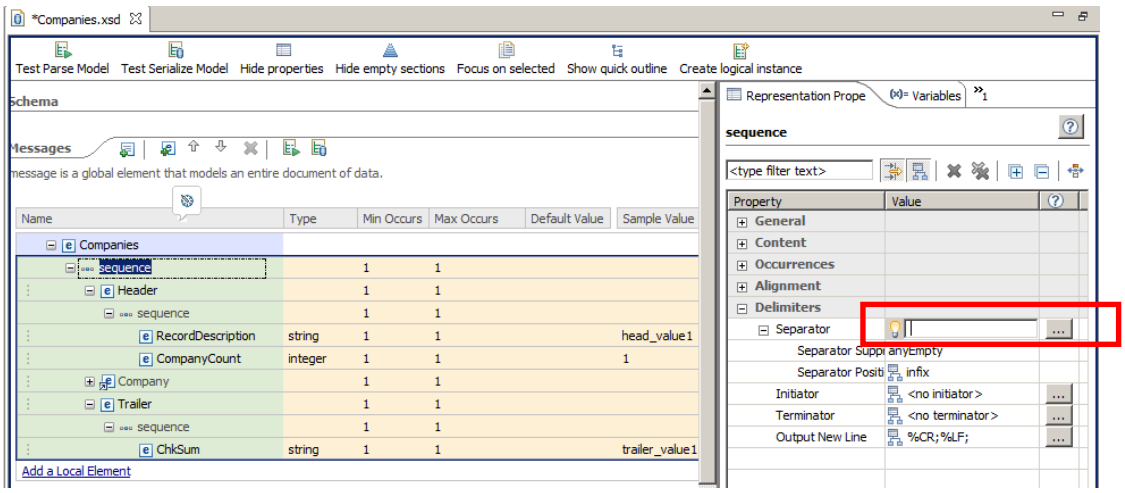


ChkSum (in Trailer):

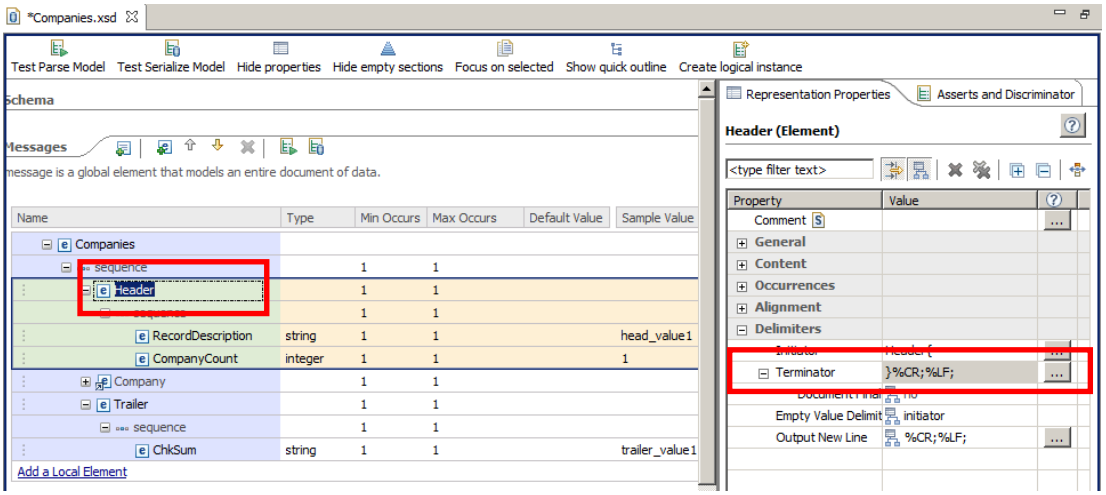


20. Now click the <sequence> content of the Companies element.

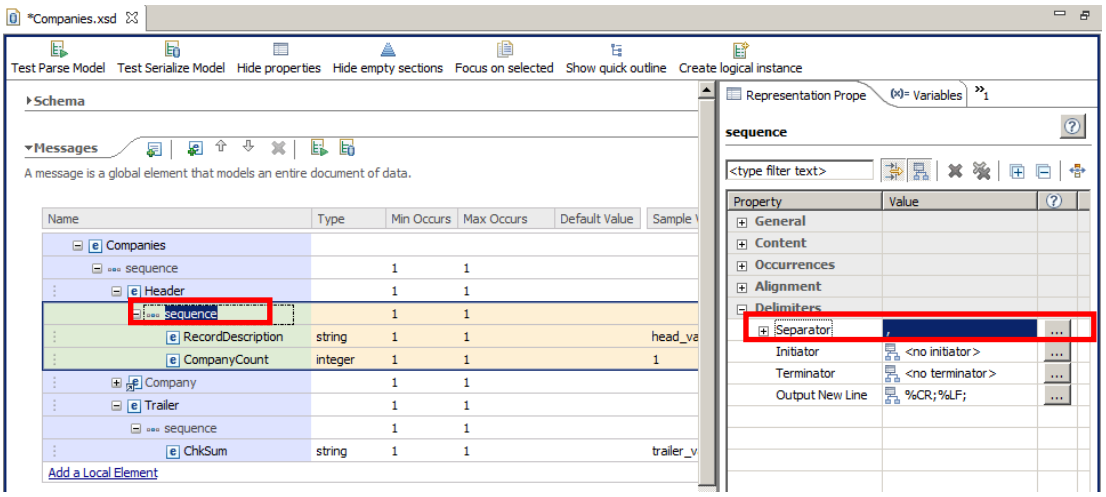
In the Representation Properties, expand the Delimiters section. Delete the value of the Separator property. Click Return to make sure the value is updated. (This is a separator automatically added by the wizard, which is not needed in this case)



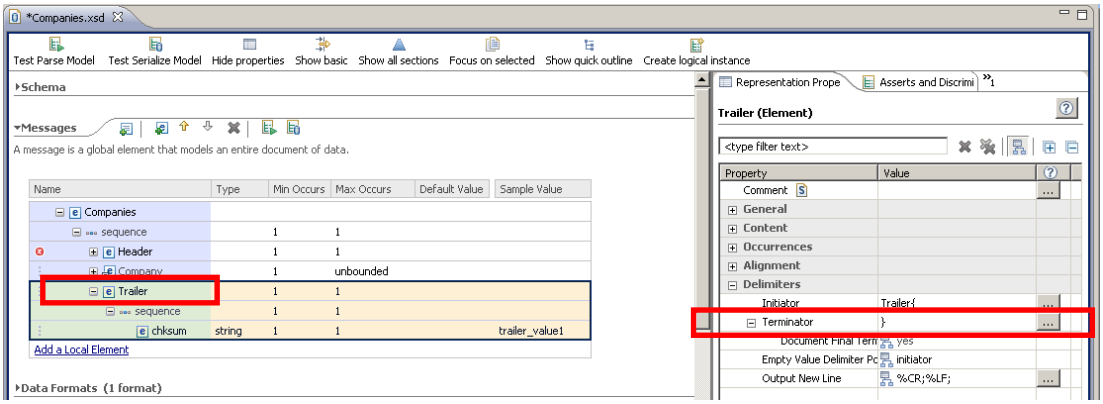
21. Click the Header element. In the Representation Properties Delimiters section, set the Terminator property to "}%CR;%LF;" (the end of the Header record is a }, followed by CRLF).



22. Then click the <sequence> content of the Header element and in the Delimiters section again, set the Separator property to "," (header elements are separated by a comma).

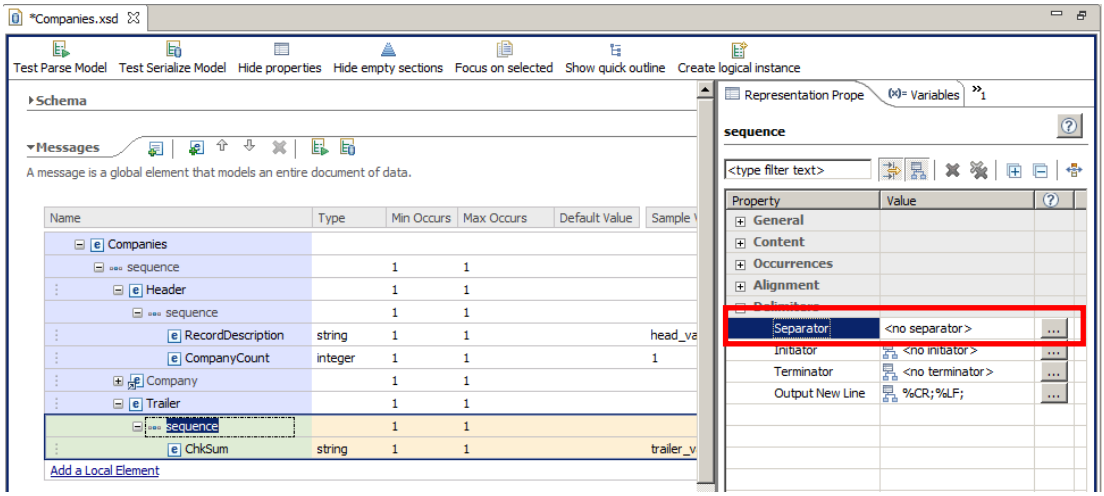


23. Click the Trailer element, and in the Delimiters section, set the Terminator property to "}".



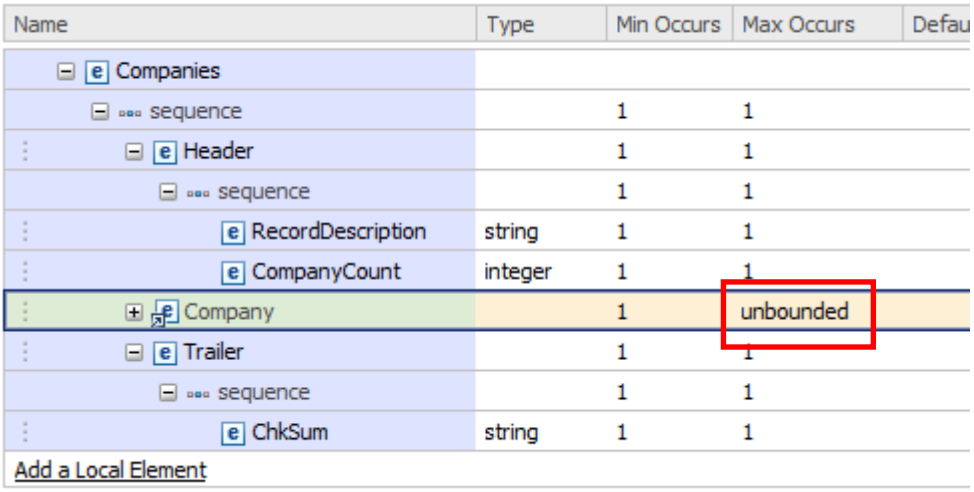
Provided by IBM BetaWorks

24. Click the <sequence> content of the Trailer element and in the Delimiters section, delete the Separator property's value.



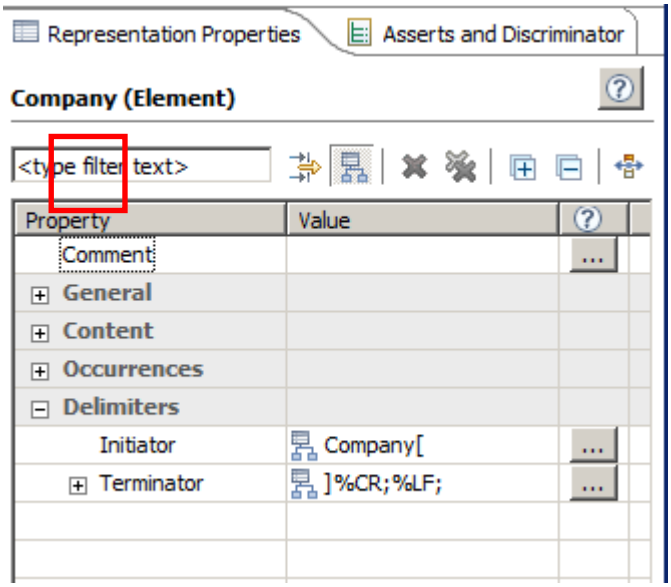
25. Click the "Max Occurs" column of the Company element reference, and change it from "1" to "unbounded".

    This will allow the Company element reference to have infinite occurrences.



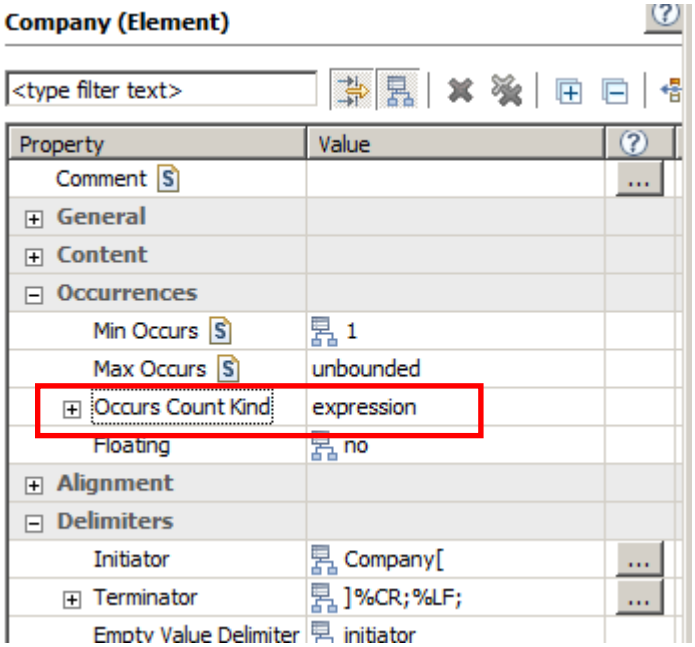
Provided by IBM BetaWorks

26. Click the Show all Properties button in the Representation Properties.



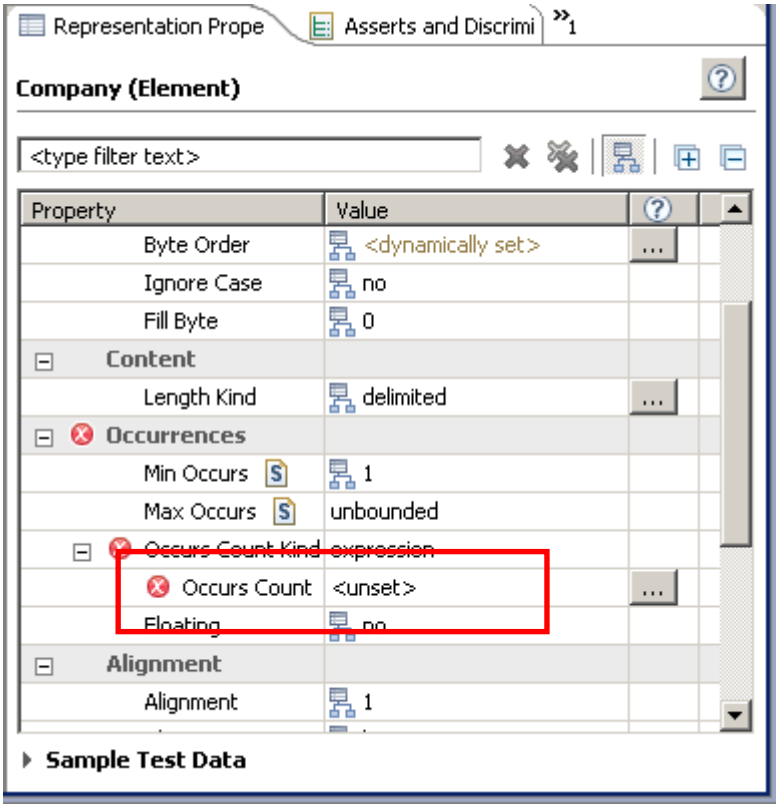
27. Click the Company element reference. In the Representation Properties Occurrences section, change the "Occurs Count Kind" from "implicit" to "expression".

    You may need to Save the model and close and reopen the schema to update the Representation Properties, so that the “Occurs Count Kind” property appears.



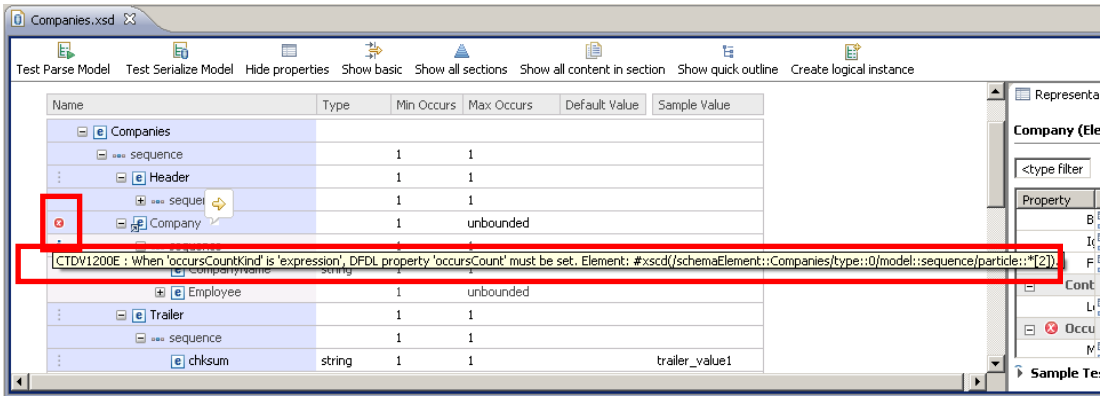
28. Expand the "Occurs Count Kind" property to show the "Occurs Count" property.

    Leave the field empty and save your XSD by pressing Ctrl+S or File->S.



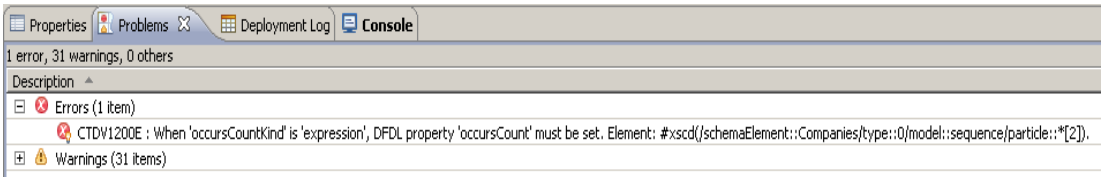
    When saving, the DFDL Editor also validates the schema file.

29. Note that an error icon has appeared next to the Company element and ‘Occurrences’.

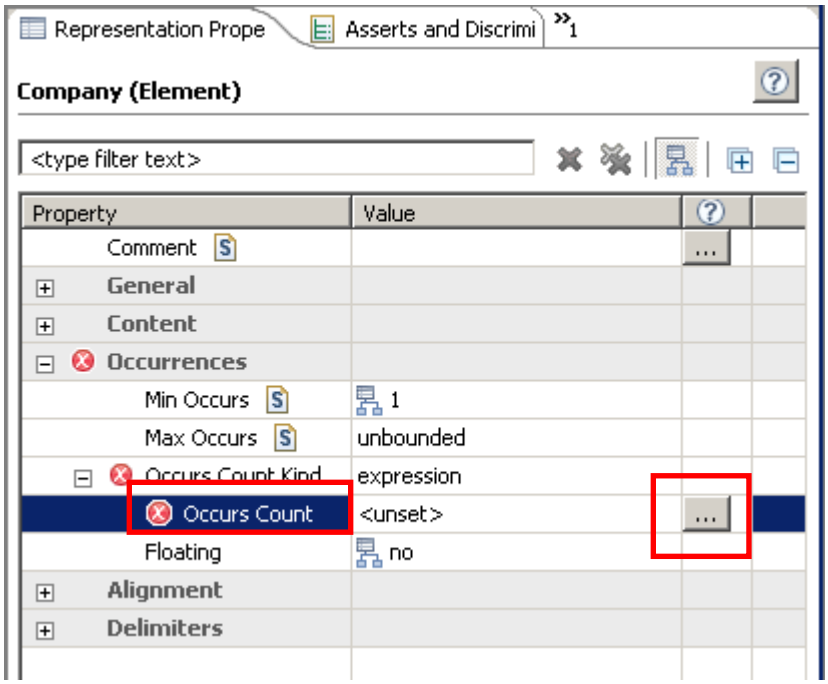


    Hover over the icon to display the error description. It states that the Occurs Count property can't be empty.

30. Click the Problems tab.

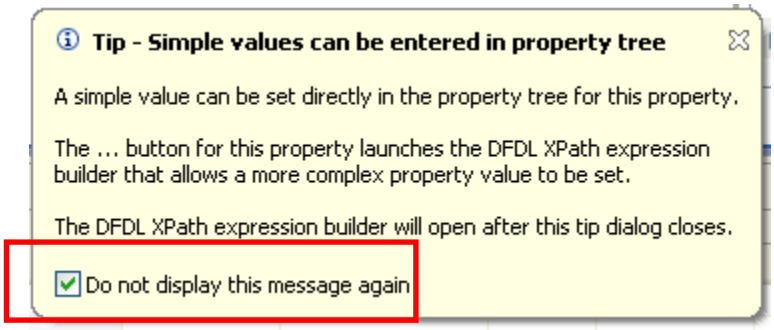


Double-click the error and the representation property of the problematic element will open.



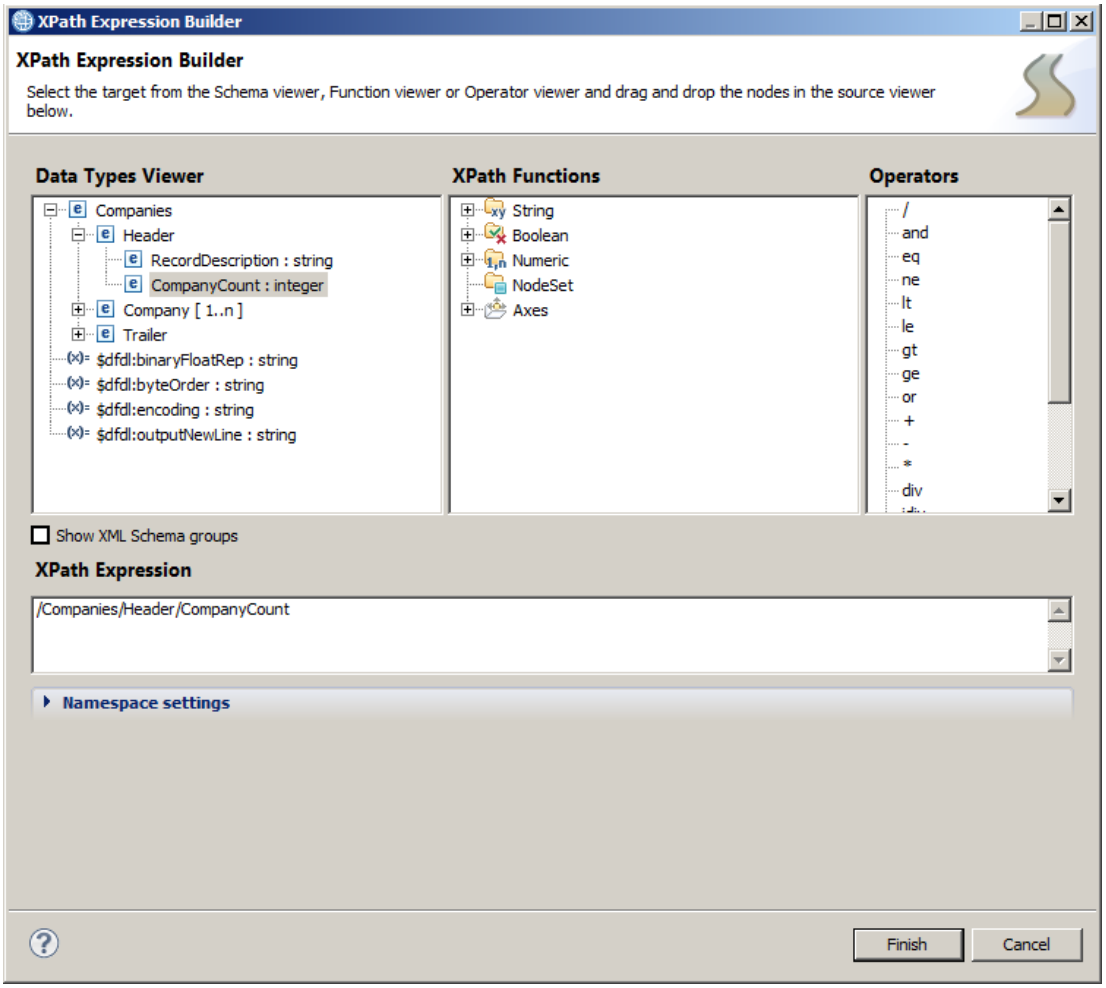
Click the button (three dots) next to the Occurs Count property.

31. A message bubble will appear. Check "Do not display this message again" and close it by clicking on the "X".



32. In the XPath Expression Builder window, expand Companies and the Header element and double-click the CompanyCount element.

    Click Finish.



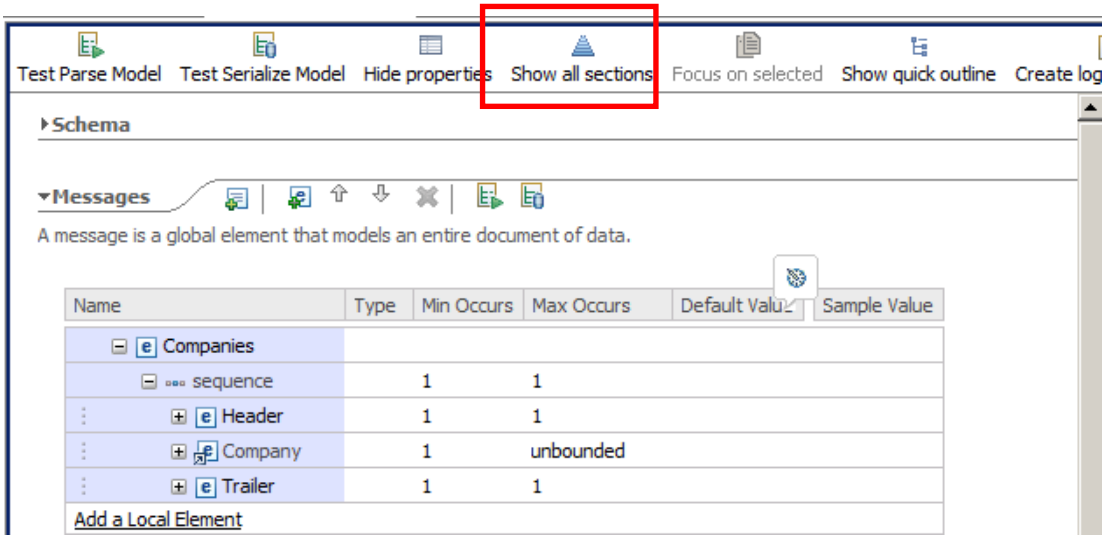
    This will indicate that the number of occurrences of the Company element is dictated by the CompanyCount element in the Header.

    Save the schema to clear the error.

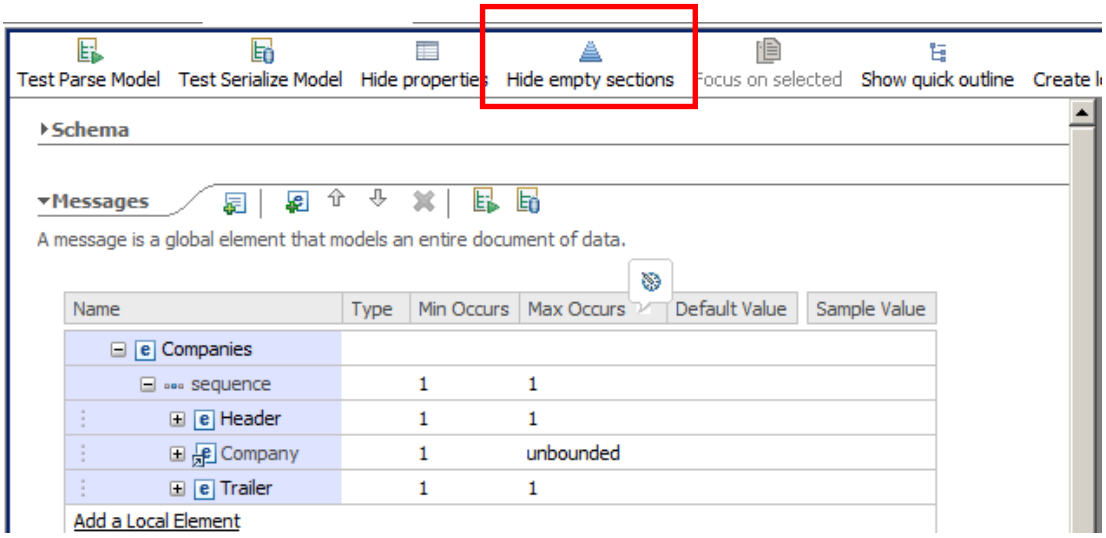
33. Finally, the input file may, or may not, have a Carriage Return Line Feed as the last character. In some cases, the last record may be missing the final “new line” character.

    To handle this situation, we will change the default property values for this model so that it will be able to handle both scenarios (ie. it will parse successfully, irrespective of whether the final character is present or not).

    In the editor, collapse the Companies model, and then click “Show all sections”.



34. To make the display a little less busy, click “Hide empty sections”.

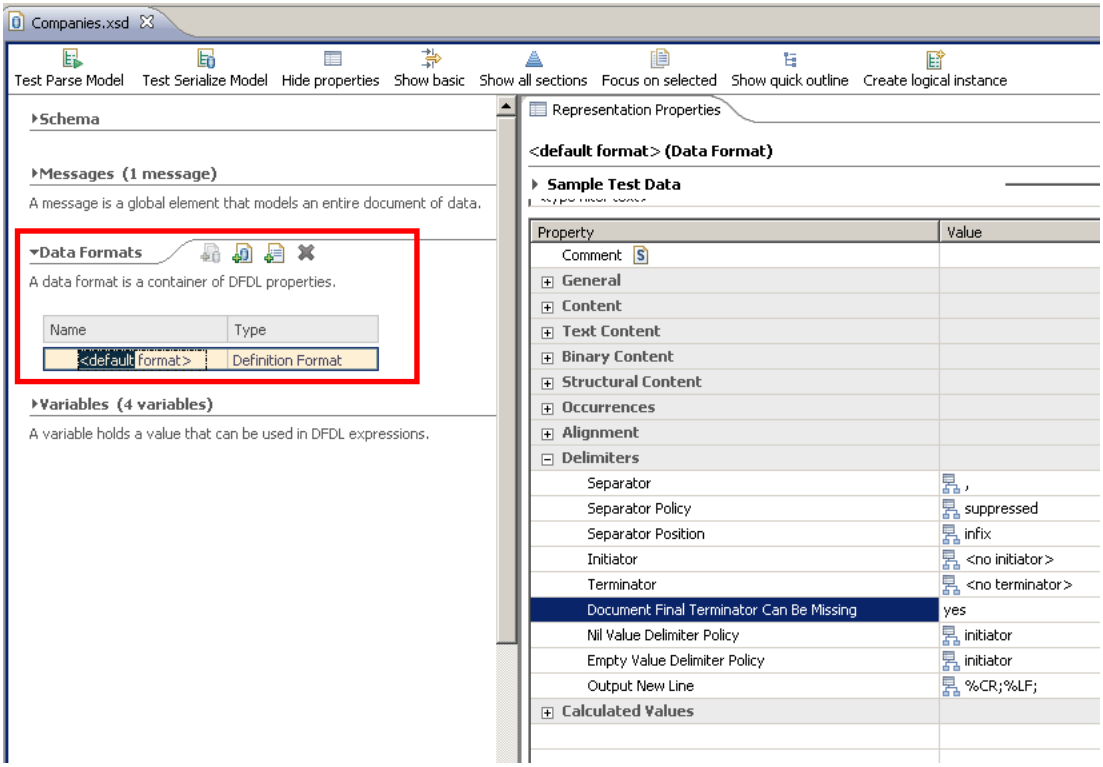


Provided by IBM BetaWorks

35. Expand the Data Formats section.

    Highlight the <default format> field. This is where you can define many default property values for the message model.

    In the Representation Properties, expand the Delimiters section, and locate the property DocumentFinalTerminatorCanBeMissing. Set this property to “yes”. Ensure you press the Return key to ensure the property is correctly updated.



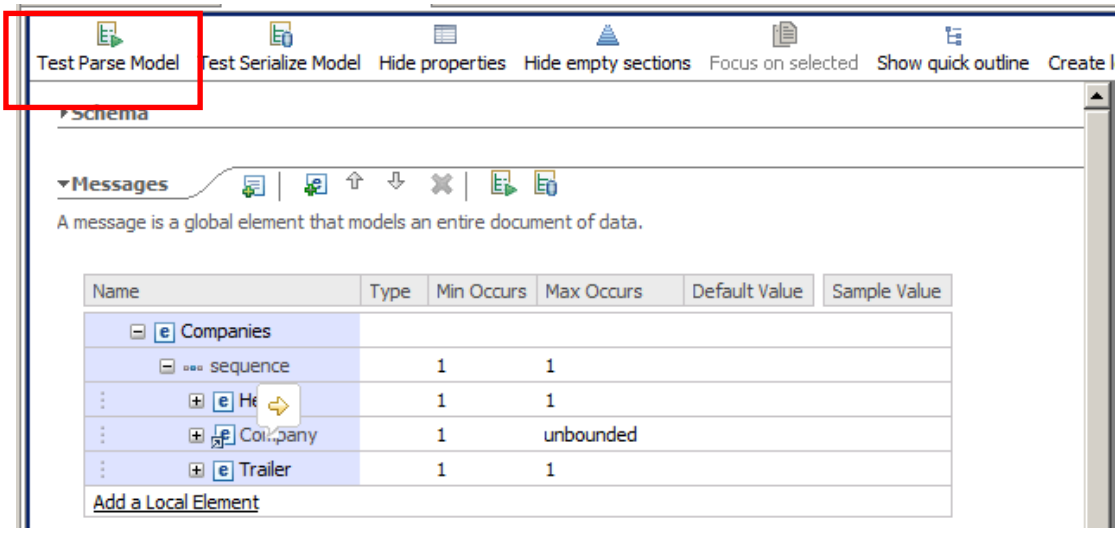
36. Press Ctrl+S or File->Save to save the schema.

    Check that the error has gone and the Problems view is clean.

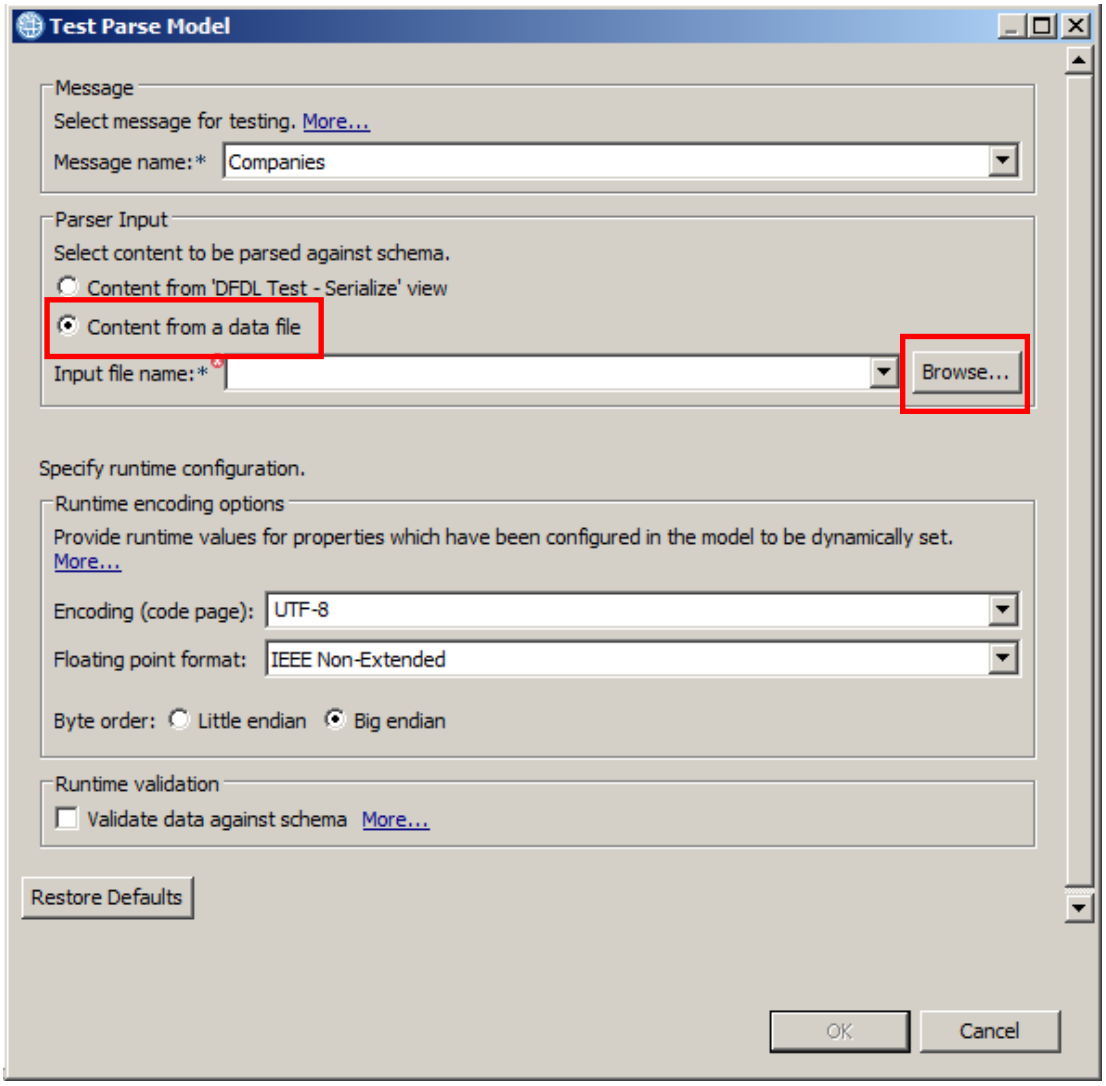
4. Testing the Message Model

1. Now that the Message Model is complete, you will test it against a delimited file.

   Click the Test Parse Model icon.

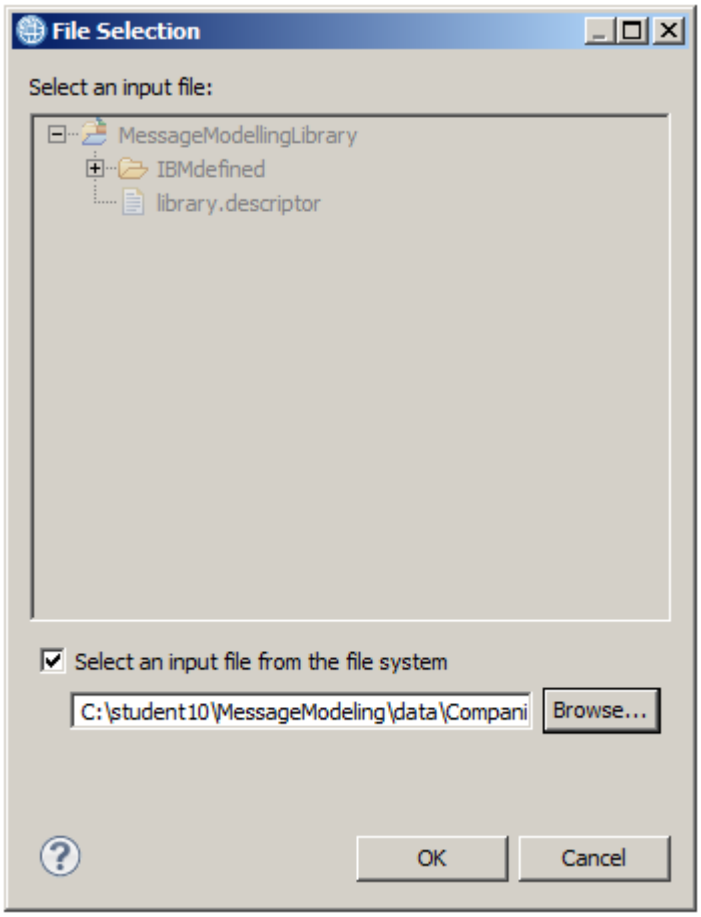


2. In the Parser Input section, select "Content from a data file" and click the Browse button.



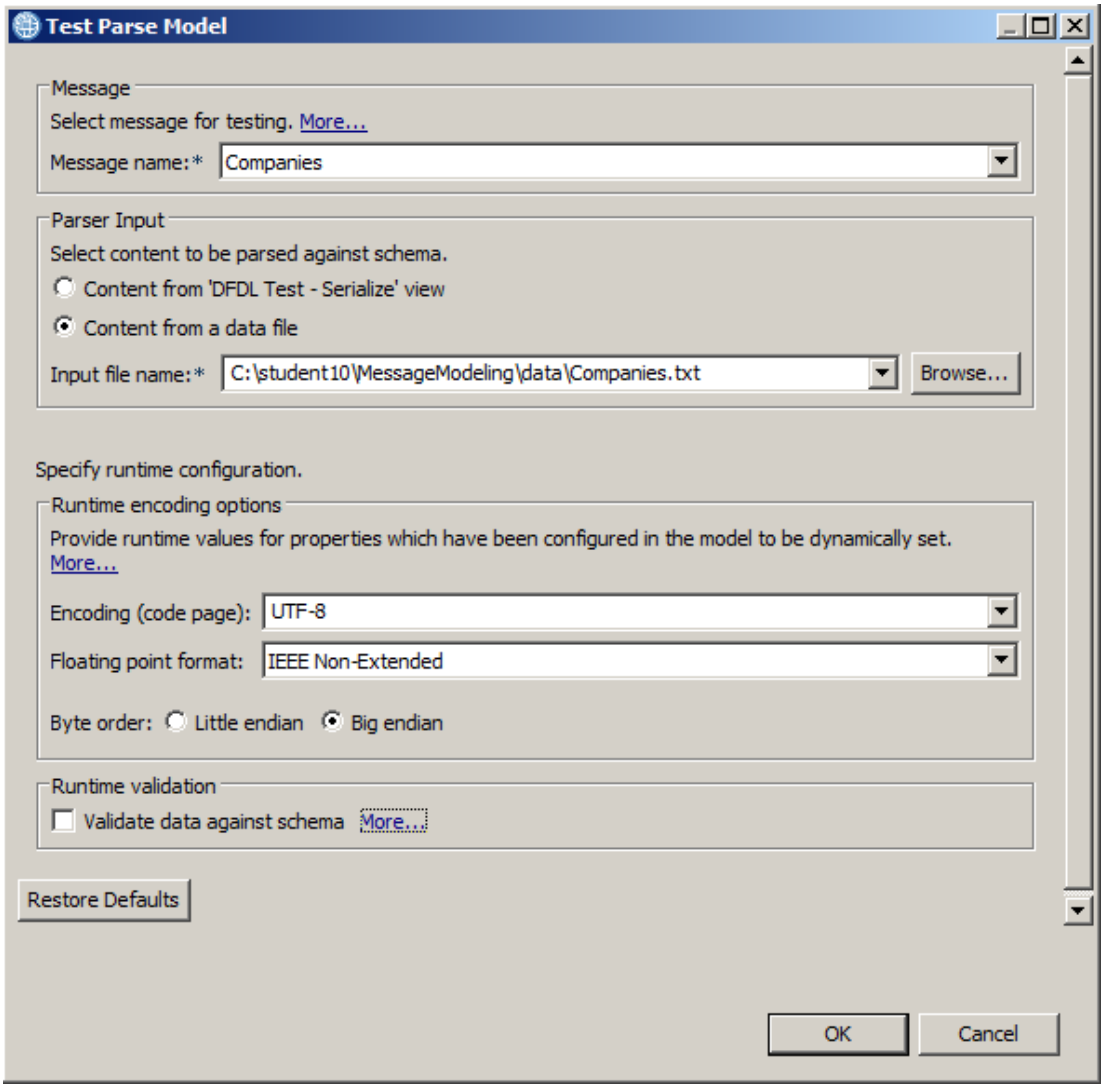
3. Check the "Select an input file from the file system" and browse to "C:\student10\MessageModeling\data" and select the "Companies.txt" file.

   Click OK.

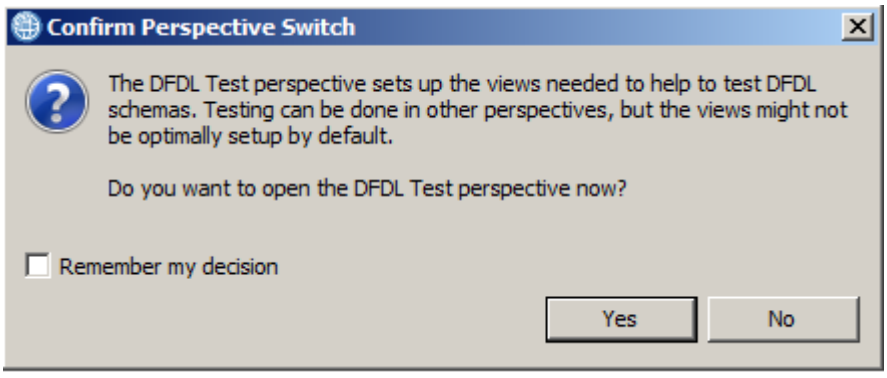


Provided by IBM BetaWorks

4. Leave all the default values and click OK.



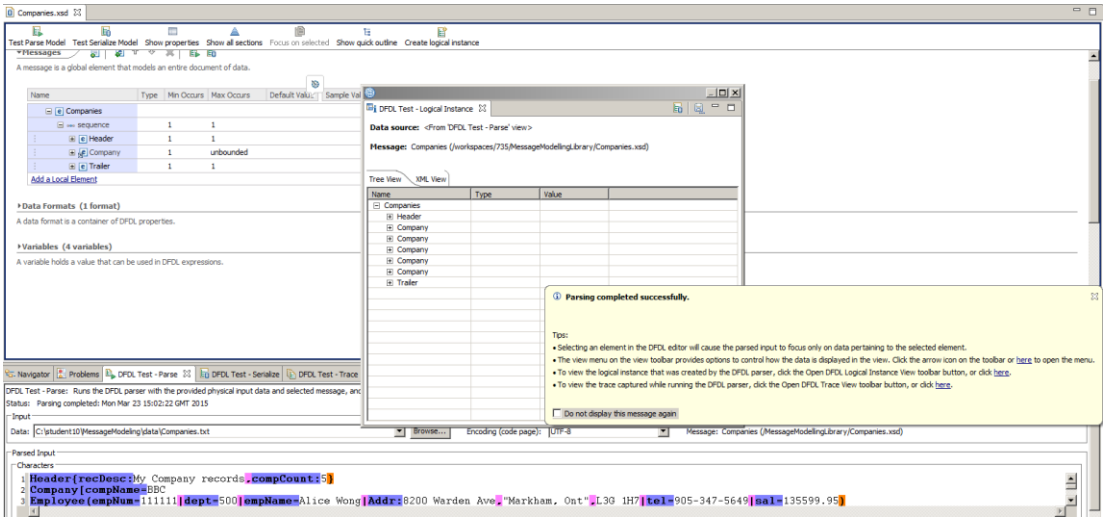
5. At the prompt, confirm the perspective switch, by clicking Yes.



Provided by IBM BetaWorks

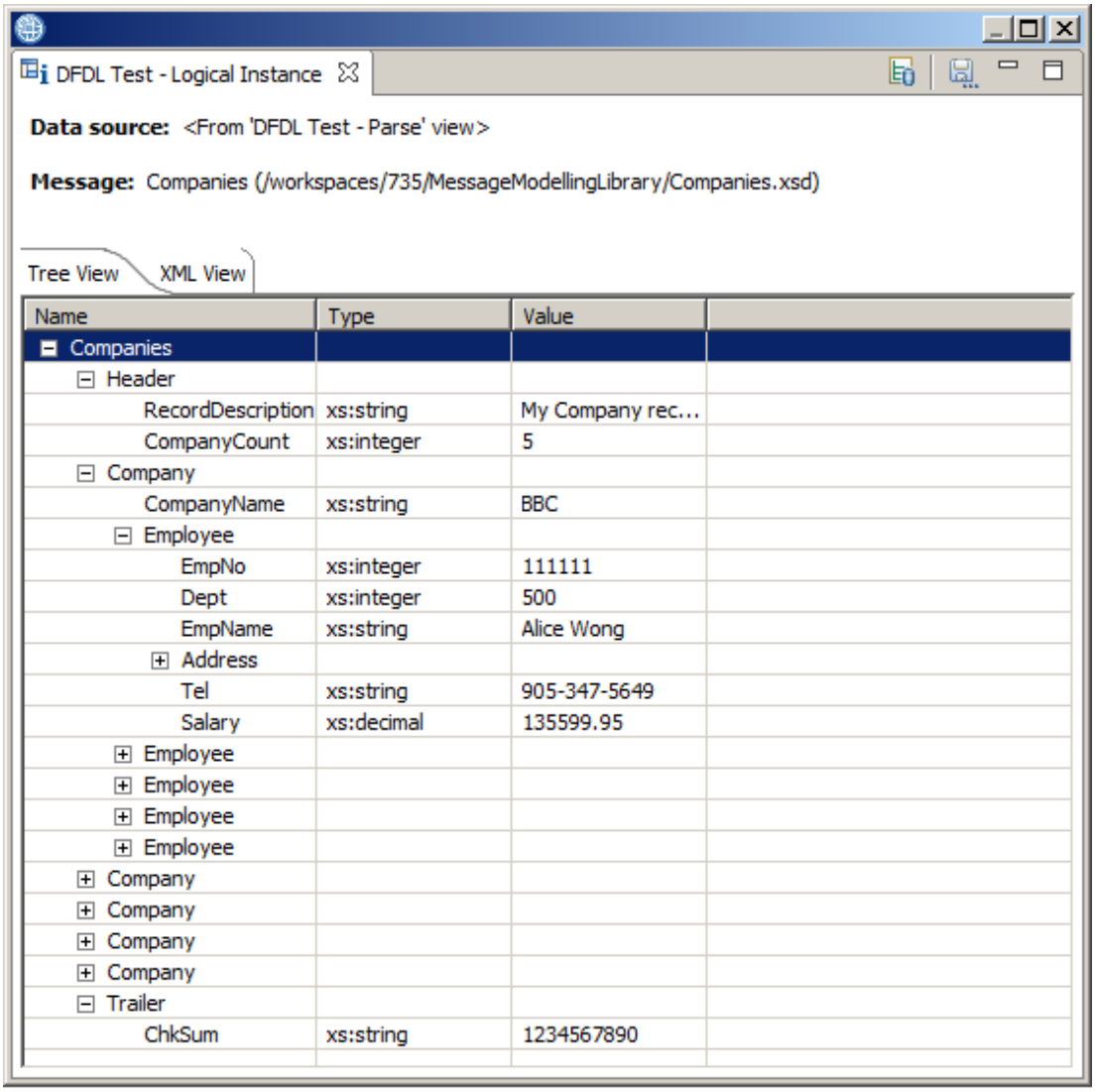
6. A message bubble with a "Parsing completed successfully" should appear.

   Check the "Do not display this message again" and close it by clicking on the "X", or click anywhere on the test parse window.



Provided by IBM BetaWorks

7. Review the DFDL Test - Parse and the Logical Instance views to verify the parsing was correct.



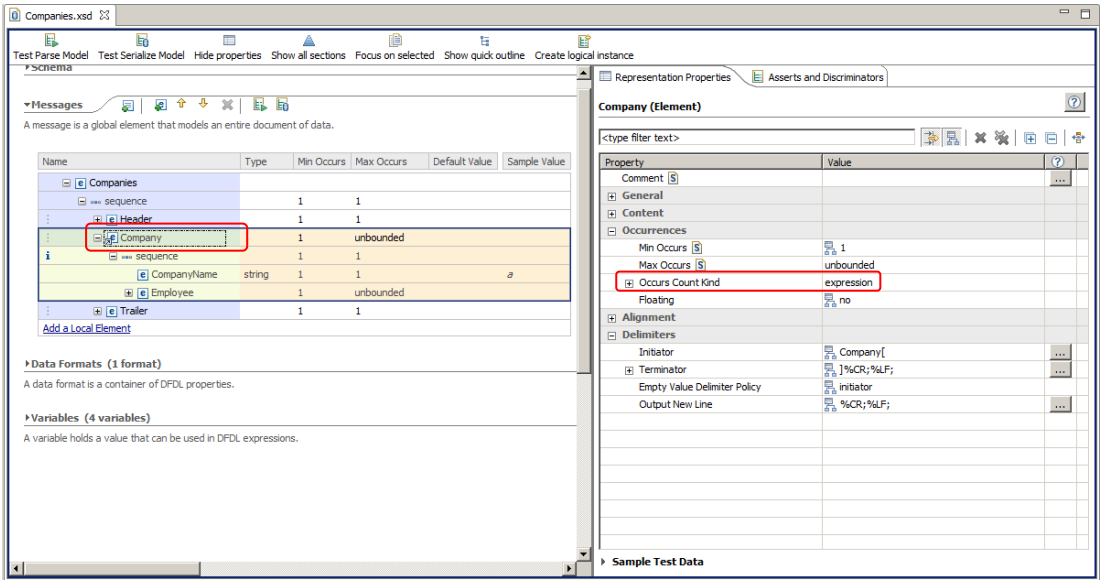
5. Establishing repeating elements by parsing

IIB V9.0.0.2 introduced the support for the DFDL facilities to enable automatic detection of repeating elements. This is provided with the "Occurs Count Kind" property. This support is enabled when this property is set to "parsed".

As a reference, the following table summarises the different values that the "Occurs Count Kind" property can take, and the meaning of each value. Note that the corresponding DFDL property name is occursCountKind.

occursCountKind value	**Meaning**	**minOccurs and maxOccurs used**
fixed	count given by maxOccurs	To provide count (minOccurs must equal maxOccurs)
expression	count given by element in data	For validation only
implicit	count bounded by minOccurs and maxOccurs	To bound count
parsed	count unbounded	For validation only

1. The Companies.xsd model has been defined to use the "Occurs Count Kind=expression" for the Company element. The value of the property is obtained from the CompanyCount element in the Header record.

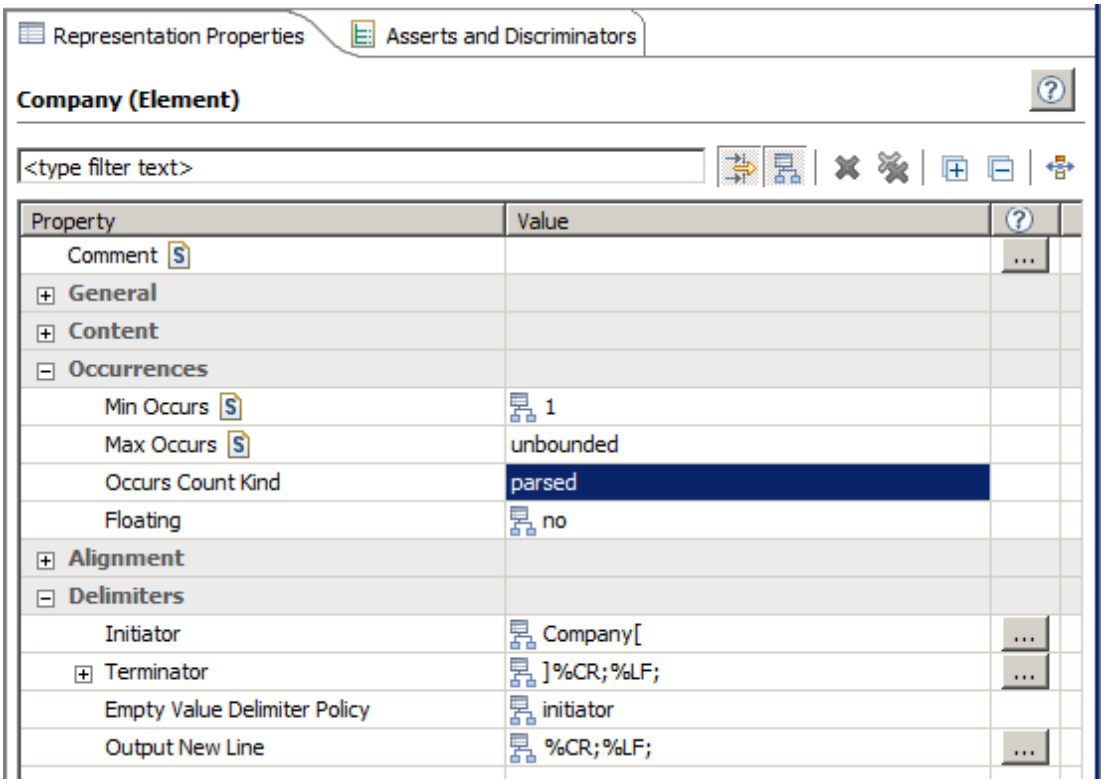


2. Change the Occurs Count Kind property to "parsed".

   This value means that the DFDL parser will keep trying to parse Company records until it gets a failure, at which point it will assume it has found all the Company records and move on to the Trailer record.

   In this case, it is no longer necessary to specify the number of records with the CompanyCount element of the header record.

   Save the updated model.



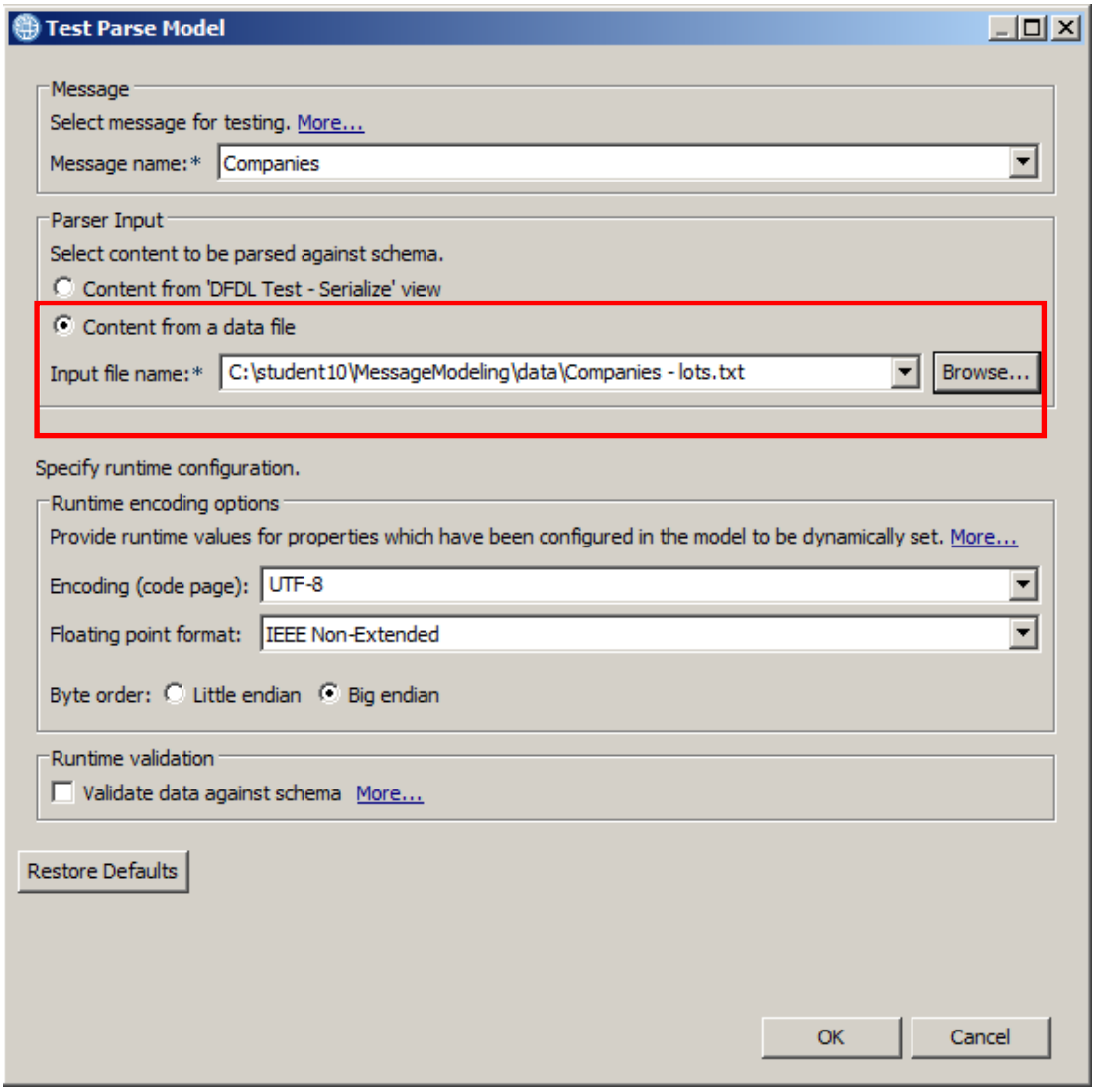
Provided by IBM BetaWorks

3. Test the updated model.

   Click Test Parse Model.

   In the dialogue, select the file **c:\student10\MessageModeling\data\Companies - lots.txt**.

   This file has more company records that the earlier input file. However, the company record count has been left unchanged, at 5, which does not match the number of company records in this new file.

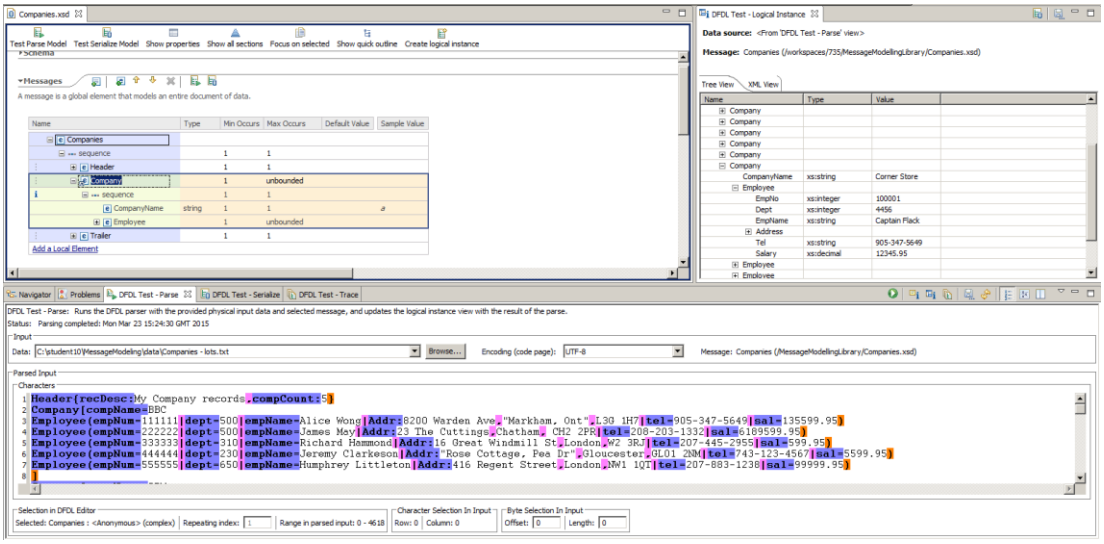


4. Click OK to invoke the Test Parser.

   The new file will be parsed, and all records will be successfully displayed, even though the CompanyCount element still has the value 5.

   The screen capture below shows the parsed infoset (top right pane) with all Company records.

   The Test Parse pane (bottom) shows the full details of all parser records. Note the value of CompanyCount (compCount) is still 5.



5. Take a quick look at the DFDL Test - Trace tab.

   Near the bottom of the trace, you will see how the parser has determined that it has found the final Company element.

   First, during processing of the 10th Company element, it looks for an Employee element. Since there are no more Employee elements, the parser determines this is not available, and tries to find a new Company element.

```
info: Offset: 4589. Starting to process element 'Employee'.
[dfdl = /MessageModelling_TDS/Companies.xsd, scd = #xscd(/schemaElement::Company/type::0/model::sequence/schemaElement::Employee), 61]

info: Offset: 4589. Did not find initiator for 'Employee'
[dfdl = /MessageModelling_TDS/Companies.xsd, scd = #xscd(/schemaElement::Company/type::0/model::sequence/schemaElement::Employee), 59]

info: Offset: 4589. Element 'Employee' is empty because the initiator was not found in the data stream.
[dfdl = /MessageModelling_TDS/Companies.xsd, scd = #xscd(/schemaElement::Company/type::0/model::sequence/schemaElement::Employee), 105]

error: CTDP3041E: Initiator 'Employee(' not found at offset '4,589'  for element '/Companies[1]/Company[10]/Employee[6]'.
```

   A little further down, it then tests for a new Company element. However, since it has processed the final Company element, the parser detects that no more are available (it could not find the initiator).

```
info: Offset: 4592. Optional element 'Company' encountered. The DFDL parser will return to this position if the e
[dfdl = /MessageModelling_TDS/Companies.xsd, scd = #xscd(/schemaElement::Company), 156]

info: Offset: 4592. Starting to process element 'Company'.
[dfdl = /MessageModelling_TDS/Companies.xsd, scd = #xscd(/schemaElement::Company), 60]

info: Offset: 4592. Did not find initiator for 'Company'
[dfdl = /MessageModelling_TDS/Companies.xsd, scd = #xscd(/schemaElement::Company), 58]

info: Offset: 4592. Element 'Company' is empty because the initiator was not found in the data stream.
[dfdl = /MessageModelling_TDS/Companies.xsd, scd = #xscd(/schemaElement::Company), 104]

error: CTDP3041E: Initiator 'Company[' not found at offset '4,592'  for element '/Companies[1]/Company[11]'.
```

So, finally, the parser decides that the 10th occurrence of Company is the last one available, and so processes the next element as a Trailer element. This is parsed successfully, using the Trailer initiator.

```
info: Offset: 4592. Parser was unable to resolve data on the current branch and will evaluate the next available b
[dfdl = /MessageModelling_TDS/Companies.xsd, scd = #xscd(/schemaElement::Companies/type::0/model::sequence), 216]

info: Offset: 4592. Occurrence '11' of element 'Company' was not found in the data. occursCountKind is 'parsed' sc
[dfdl = /MessageModelling_TDS/Companies.xsd, scd = #xscd(/schemaElement::Company), 171]

info: Offset: 4592. Starting to process element 'Trailer'.
[dfdl = /MessageModelling_TDS/Companies.xsd, scd = #xscd(/schemaElement::Companies/type::0/model::sequence/schemaE

info: Offset: 4592. Found initiator 'Trailer{' for 'Trailer'
[dfdl = /MessageModelling_TDS/Companies.xsd, scd = #xscd(/schemaElement::Companies/type::0/model::sequence/schemaE

info: Offset: 4600. Starting to process element 'ChkSum'.
[dfdl = /MessageModelling_TDS/Companies.xsd, scd = #xscd(/schemaElement::Companies/type::0/model::sequence/schemaE

info: Offset: 4600. Found initiator 'chksum:' for 'ChkSum'
[dfdl = /MessageModelling_TDS/Companies.xsd, scd = #xscd(/schemaElement::Companies/type::0/model::sequence/schemaE

info: Offset: 4607. Found delimited value: '1234567890' for element 'ChkSum'. The delimiter was '}'.
[dfdl = /MessageModelling_TDS/Companies.xsd, scd = #xscd(/schemaElement::Companies/type::0/model::sequence/schemaE
```


Provided by IBM BetaWorks

6. Unordered sequences

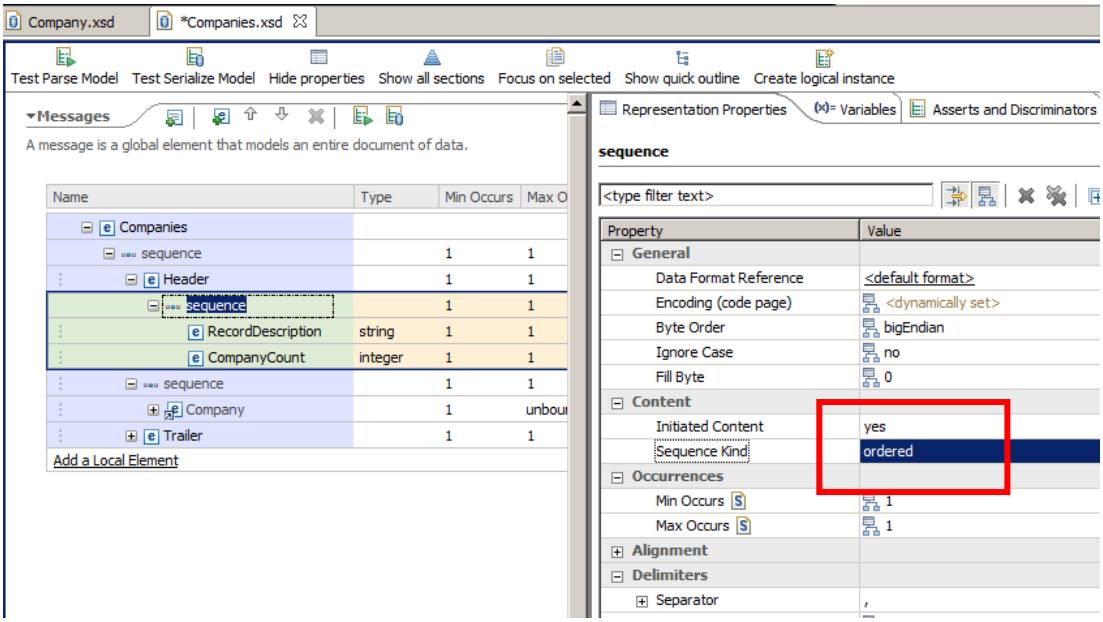
Finally, you will examine the facilities for handling unordered sequences. This was also introduced in IIB Version 9.0.0.2

This facility will be demonstrated by switching the order of two elements in the Header record of the Companies.txt file, and by setting the appropriate property in the model to use "unordered".

1. Switch back to the Application Development perspective.

   In the Companies.xsd model, expand the Header element, and highlight the "sequence" underneath Header (ie. the sequence referring to the two elements contained within Header).

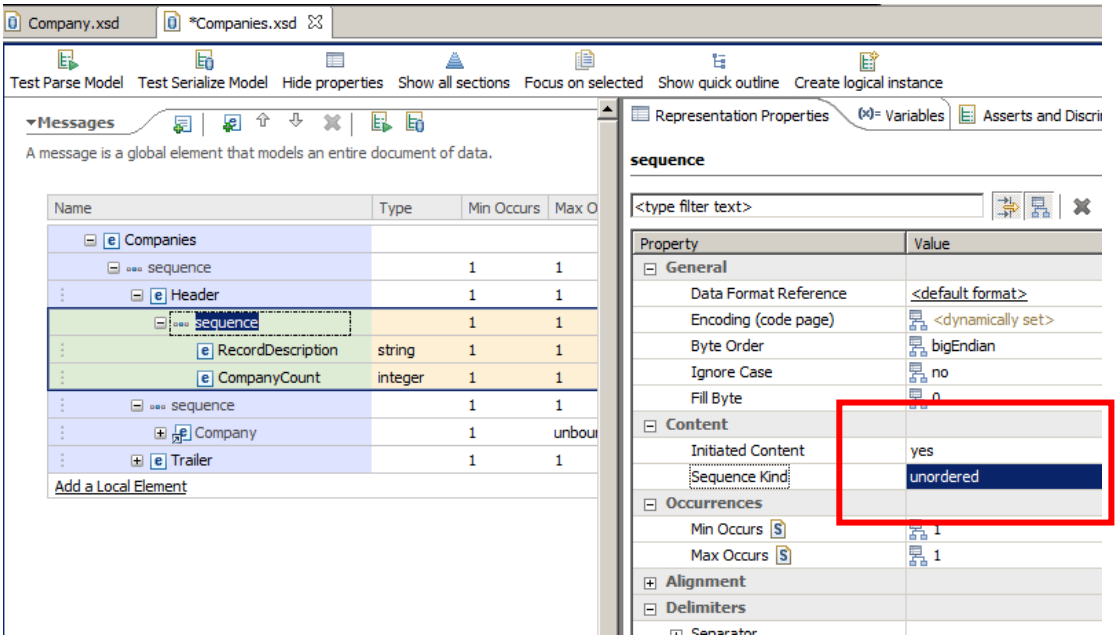
   In the Representation Properties (Content), observe that the Sequence Kind property is set to "ordered".



Provided by IBM BetaWorks

2. Set the property to "unordered", and save the model.

   Note that you are not changing the order of the elements in the model definition.



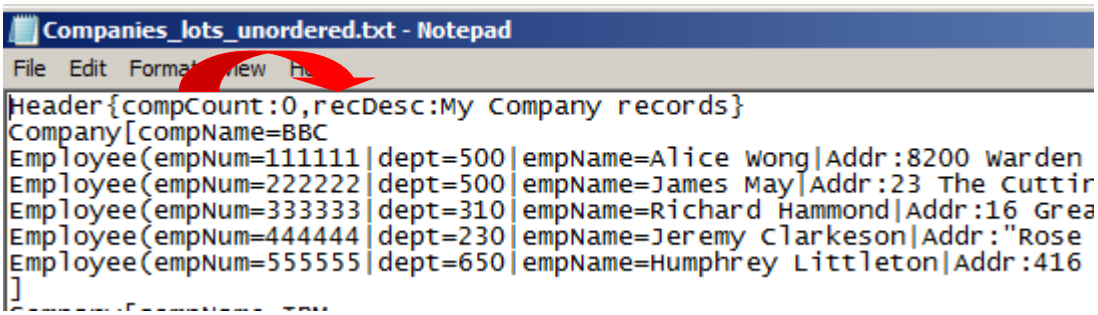
Provided by IBM BetaWorks

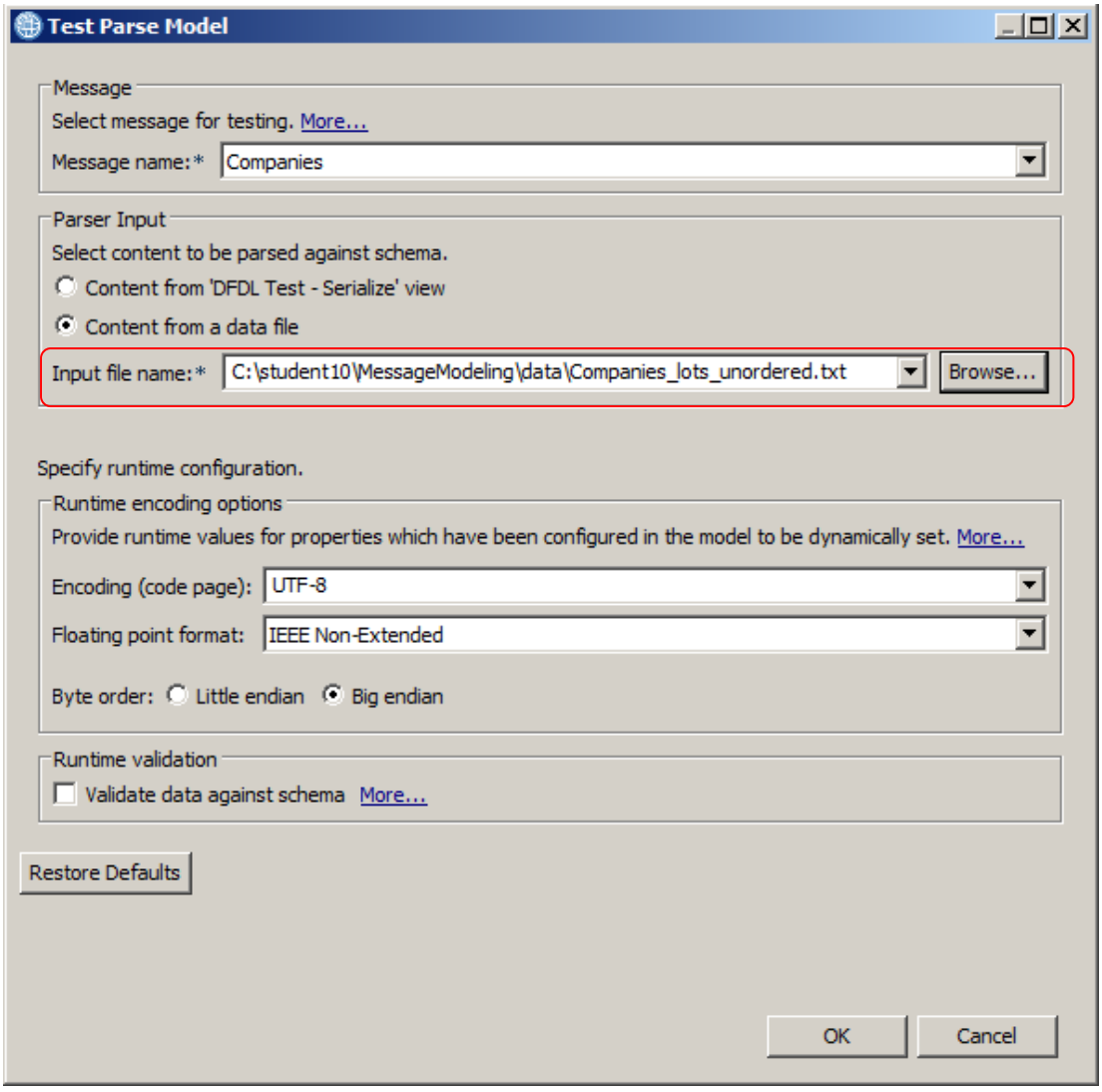
3. Test the updated model again.

   Click Test Parse Model.

   In this case, select the file
   c:\student10\MessageModeling\data\Companies_lots_unordered.txt.

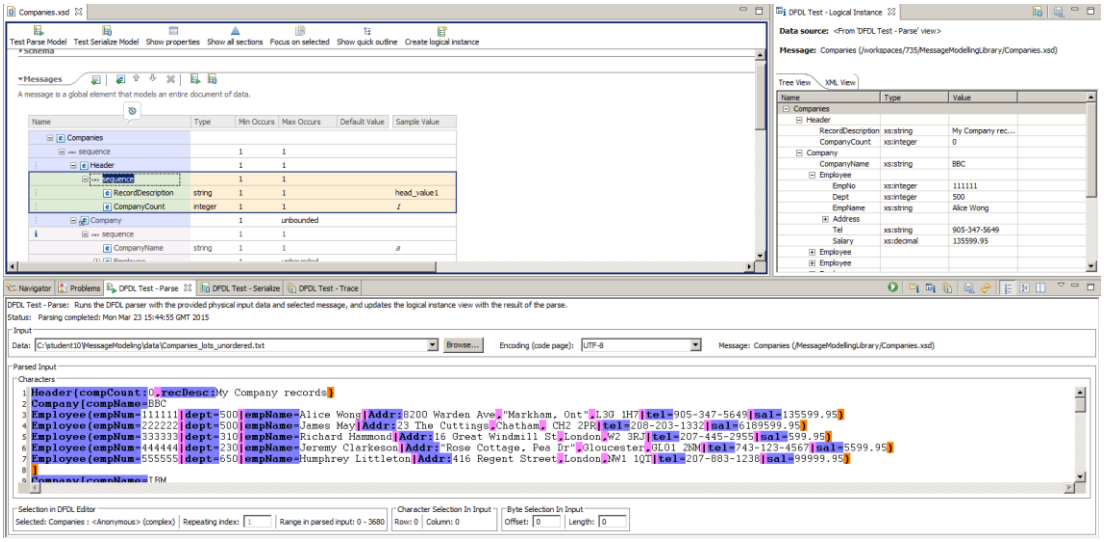
   This file has a lot of company records. In addition, the elements in the Header record have been switched, so that the compCount element appears before the recDesc.





   Select OK.

4. You will see that the new file has been successfully parsed, even though the Header elements in the input message have been switched round.



END OF LAB GUIDE