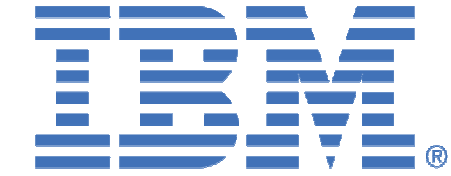


IBM Integration Bus

Message Modeling with DFDL

Lab 5
Using DFDL length prefixes

June 2015
Hands-on lab built at product
Version 10.0.0.0

1. INTRODUCTION 3

1.1 LAB PREPARATION 3
1.2 LAB SCENARIO 3

2. IMPORT THE BASE MODEL 4

3. CREATE THE PREFIX LENGTH CHARACTER SCENARIO 8

4. CREATE THE PREFIX LENGTH BINARY SCENARIO 20

END OF LAB GUIDE 28

Provided by IBM BetaWorks

1. Introduction

Support for length prefixes in the DFDL Message Modelling tools was introduced in WebSphere Message Broker V8.0.0.1 and is included in IBM Integration Bus V9.0 and V10.

A common form of data formatting uses the approach of having a prefix to the main element, where the prefix contains the length of the element itself. This capability is commonly used in message modeling, and is a particular requirement for certain types of industry standard models, for example the ISO8583 standard used in credit card processing, and the PL/1 var char type.

There are many variations of this approach. The value held in the length prefix might represent just the length of the element to which it refers, or the value in the length prefix might include the length of the prefix as well as that of the element. The length prefix itself might have different characteristics from the element, for example it may be a binary prefix whereas the element is text. It is even possible for a length prefix to have its own length provided by another length prefix!

This lab will illustrate some of these variations of length prefix specifications.

1.1 Lab preparation

To run this lab, unzip the supplied file MessageModelling.zip into the directory c:\student10 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 extends the Tagged / delimited lab, and includes the new message modeling capability for prefix length fields introduced in WMB V8 Fixpack1.

The starting point for this lab is a tagged-delimited message model, with a schema definition named Company.xsd. You will create two new message models based on this, as follows:

CompanyAddressChar.xsd – some of the elements will be changed to use a 2-byte length prefix of type “character”.

CompanyAddressBin.xsd – some of the elements will be changed to use a 2-byte length prefix of type “binary”.

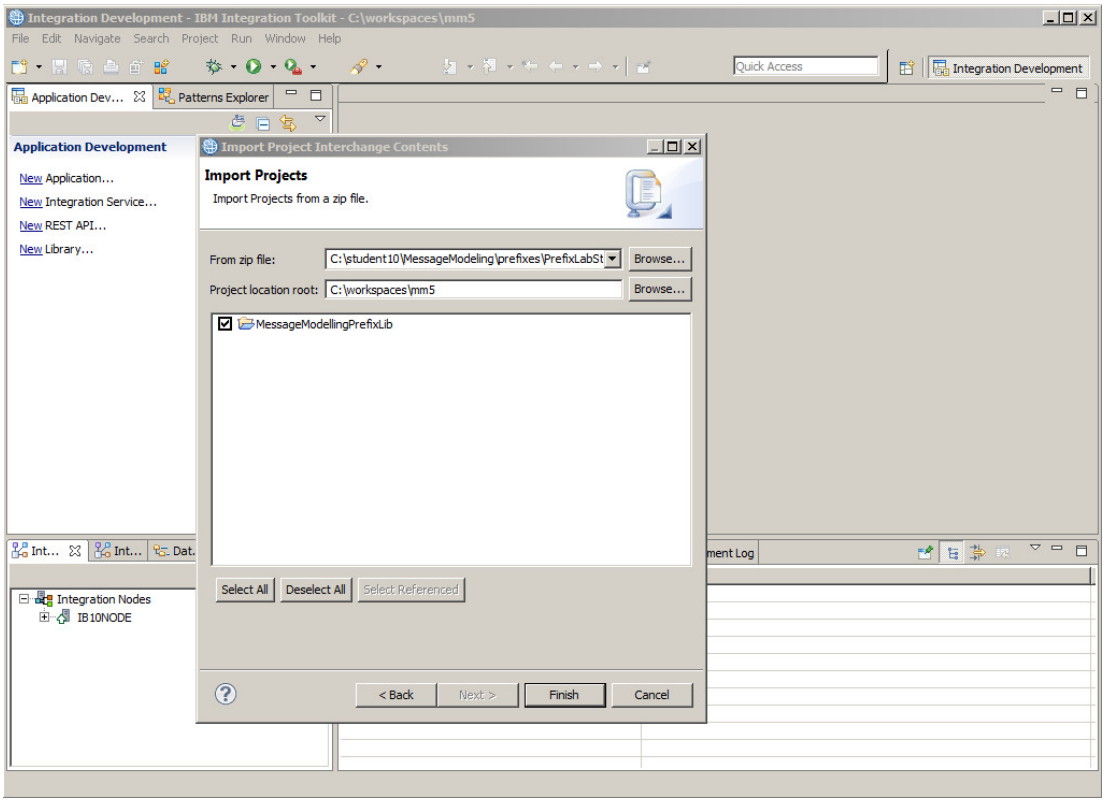
Provided by IBM BetaWorks

2. Import the base model

You are going to create two message models. One will use a length prefix in character form, and one will use a length prefix in binary form. Both length prefixes will be two bytes.

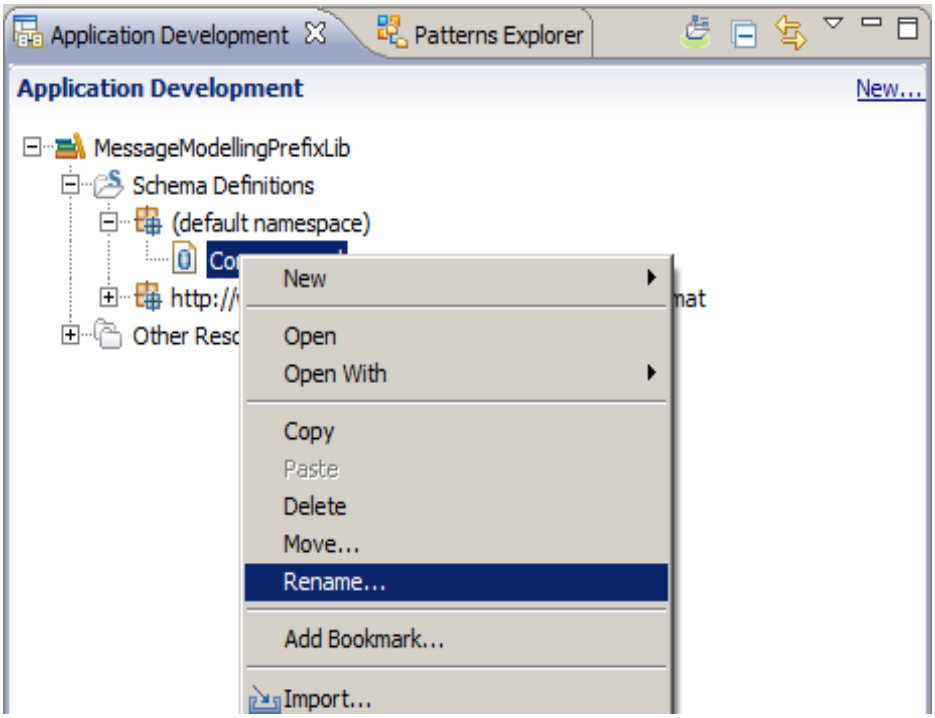
Both message models will be defined in the same library, so you will need to make various adjustments to the schema and message names to avoid naming conflicts.

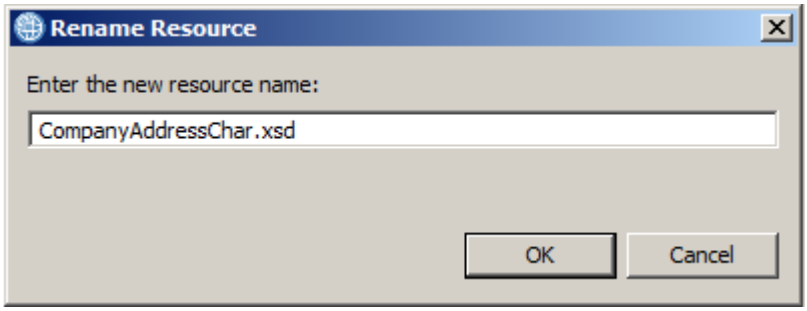
1. Import the PI file c:\student10\MessageModelling\prefixes\PrefixLabStartingPoint.zip.



Provided by IBM BetaWorks

2. Rename the schema Company.xsd to CompanyAddressChar.xsd.

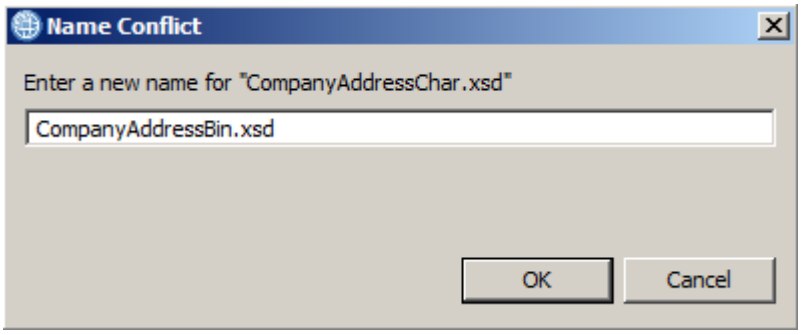




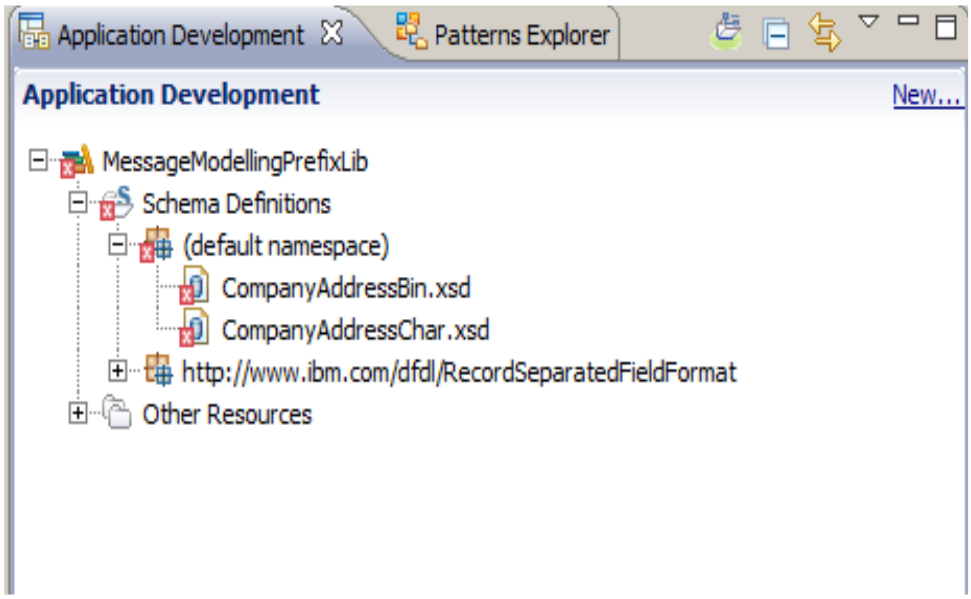
Provided by IBM BetaWorks

3. Create a new copy of the schema, and call it CompanyAddressBin.xsd.

   Use Ctrl-C / Ctrl-V.

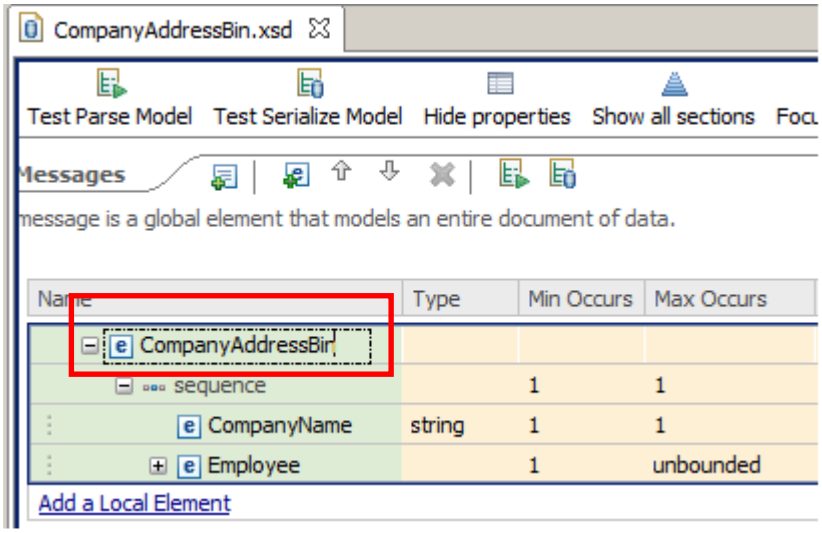


4. At this point, the navigator will show several errors. This is because the two models have a global element with the same name, which is not permitted within a single library.



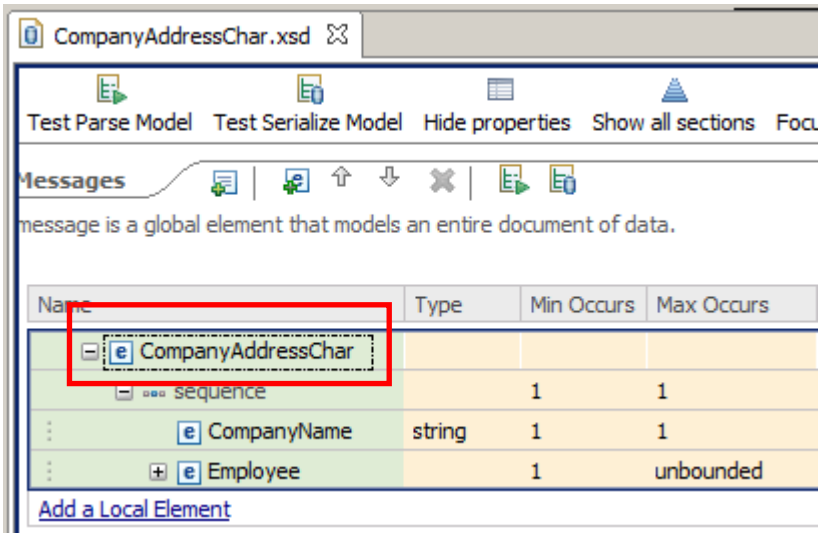
Provided by IBM BetaWorks

5. To rectify this, open the CompanyAddressBin.xsd, and in the message model editor, change the global element name to CompanyAddressBin.



6. Saving this change (Ctrl-S) will remove the errors.

   For consistency, make a similar change to the second schema, CompanyAddressChar.xsd, renaming the global element to CompanyAddressChar.

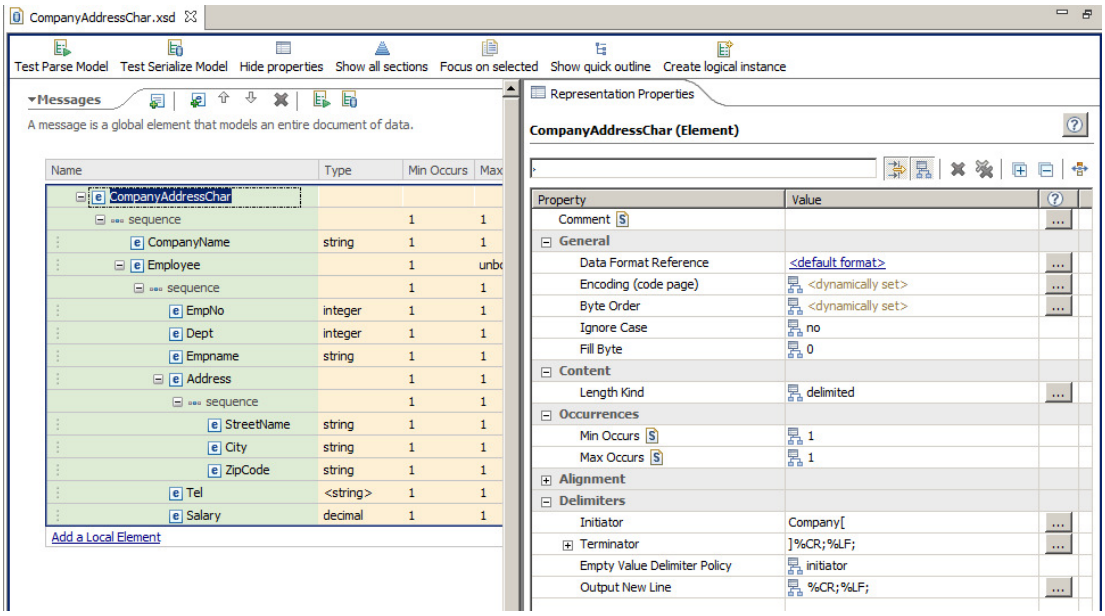


   You will now have two message models in the library, with different global elements. You are now ready to define the length prefixes.

Provided by IBM BetaWorks

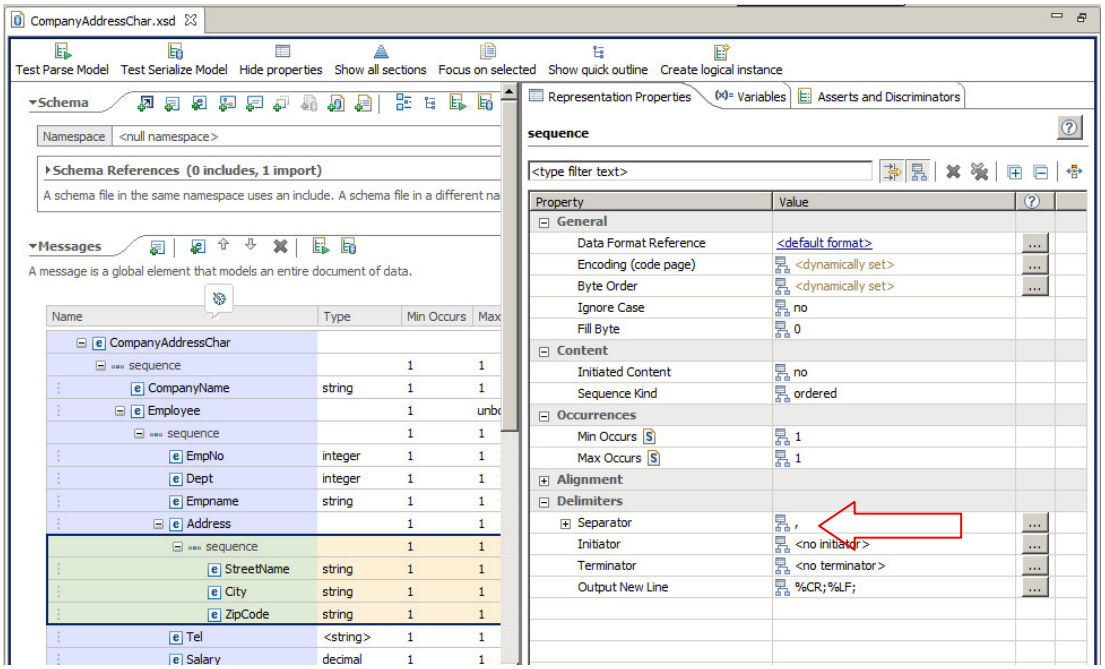
3. Create the Prefix Length Character scenario

Open and expand the CompanyAddressChar.xsd message model.



2. Highlight the Address sequence element. You will see that the separator has been set to ‘,’ (comma); this means that all fields in the Address element are separated by commas.

This is the part of the model that we will change.



3. You will change the elements in the Address global element to be identified and parsed by using length prefixes, instead of being comma-delimited.

   In this model, the length prefix is a two-character text number.

   In this case the Address global element may have a value something like this:

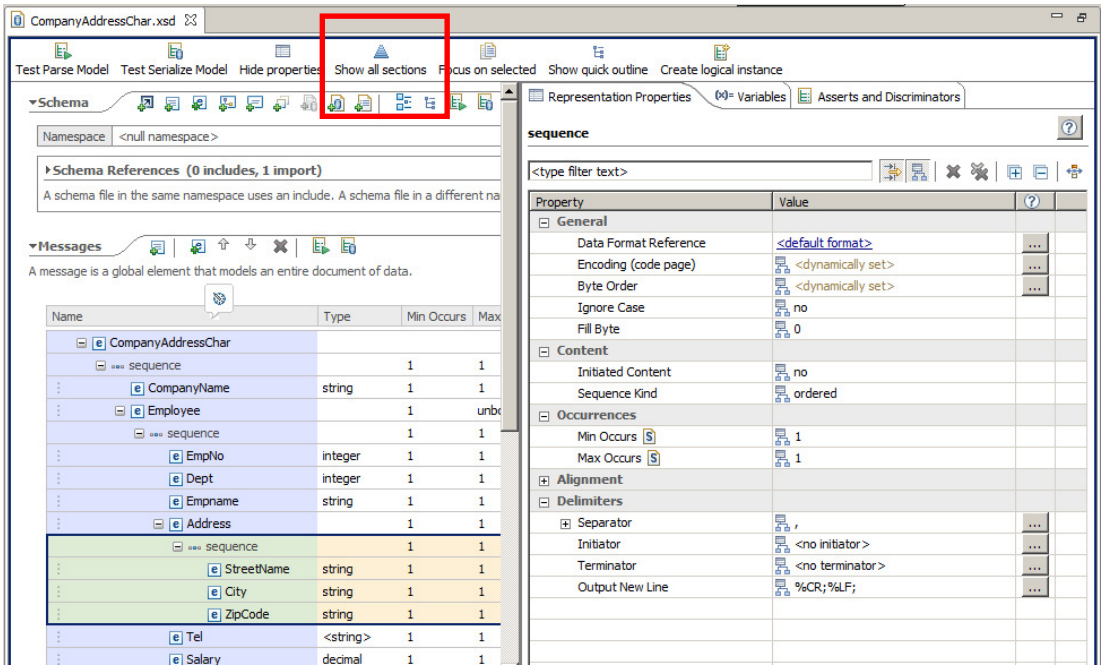
   ```
   Addr:158200 Warden Ave14"Markham, Ont"07L3G 1H7
   ```

   The StreetName field has a value of '8200 Warden Ave', and has a prefix length of 15.
   The City field has a value of '"Markham, Ont"', and has a prefix length of 14.
   The ZipCode field has a value of 'L3G 1H7', and has a prefix length of 07.

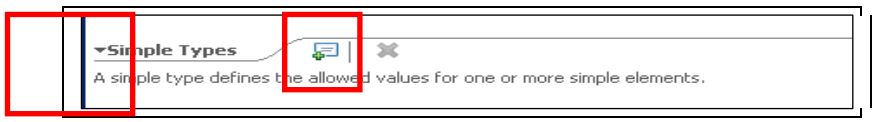
   Note that the prefix length values are normal display characters, and hence can be read in clear text.

4. To define this type of model, you first need to define a Simple Type. This is used to define the physical characteristics of the prefix length. An element which has a prefix length then simply refers to the simple type.

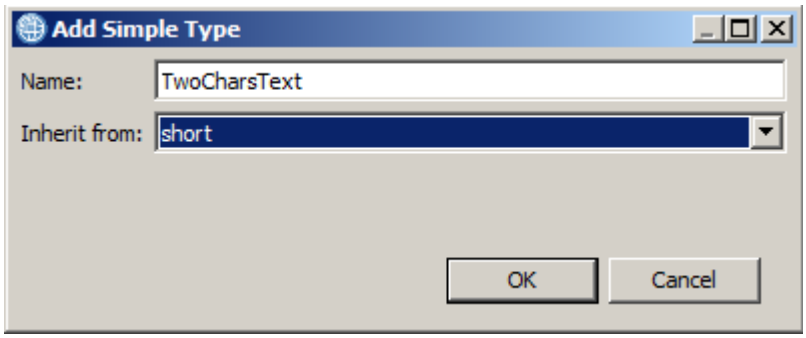
   Click "Show all sections" on the main editor line.



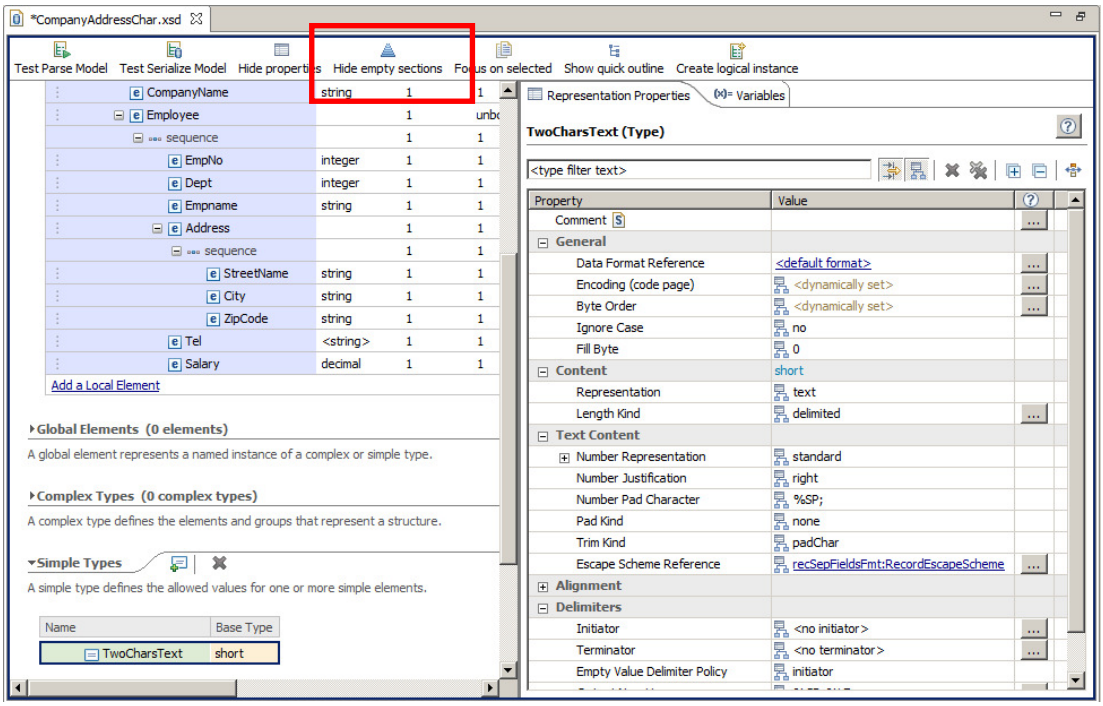
5. In the main editor pane, expand Simple Types, and then click the “Add Simple Type” button.



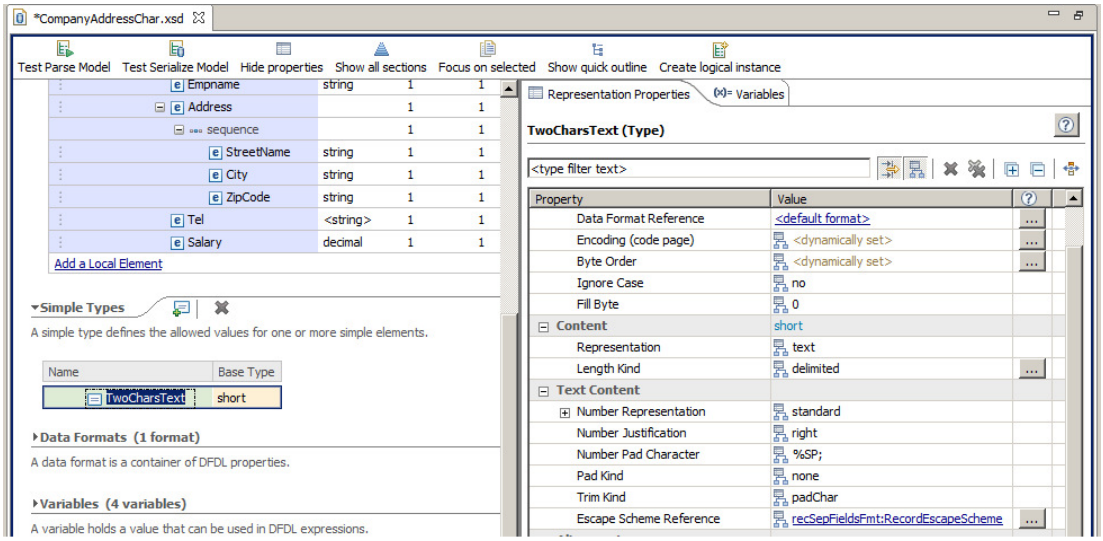
6. In the dialogue window, set Name = TwoCharsText (you can define your own descriptive name for this type), and set “Inherit from” to “short”.
Click OK.



7. To make the editor clearer, click “Hide empty sections”.



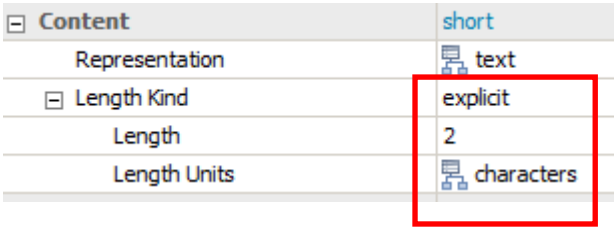
8. Highlight the new Simple Type, TwoCharsText. You will see that various properties have been set for this new type, shown in the Representation Properties in the right hand pane. Some of these properties must be changed to reflect the nature of our prefix length values.



9. First, the Content Representation has been set to “text”. This is the correct value for this scenario.

   Second, the “Length kind” is set to “delimited”. Change this to “explicit”.

   The editor will then provide two further properties. Set Length to 2, and leave Length Units as “characters”.



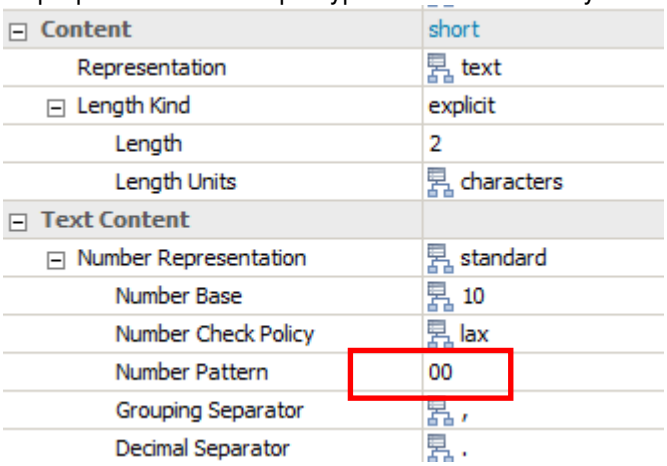
   Note that changing lengthKind from 'delimited' to 'explicit' does not necessarily mean there is no delimiter present, it means that the parser does not scan for the delimiter to establish the length.

Provided by IBM BetaWorks

10. Finally, when the number representation is “text”, the “Number Pattern” must have a defined value (it will be set to <unset>).

    In the “number pattern” field, type ‘00’ (without the quotation marks), and click return. (You can also use the wizard button for more complex patterns, but not required in this case).

    All other text number properties of the simple type can be left as they are.



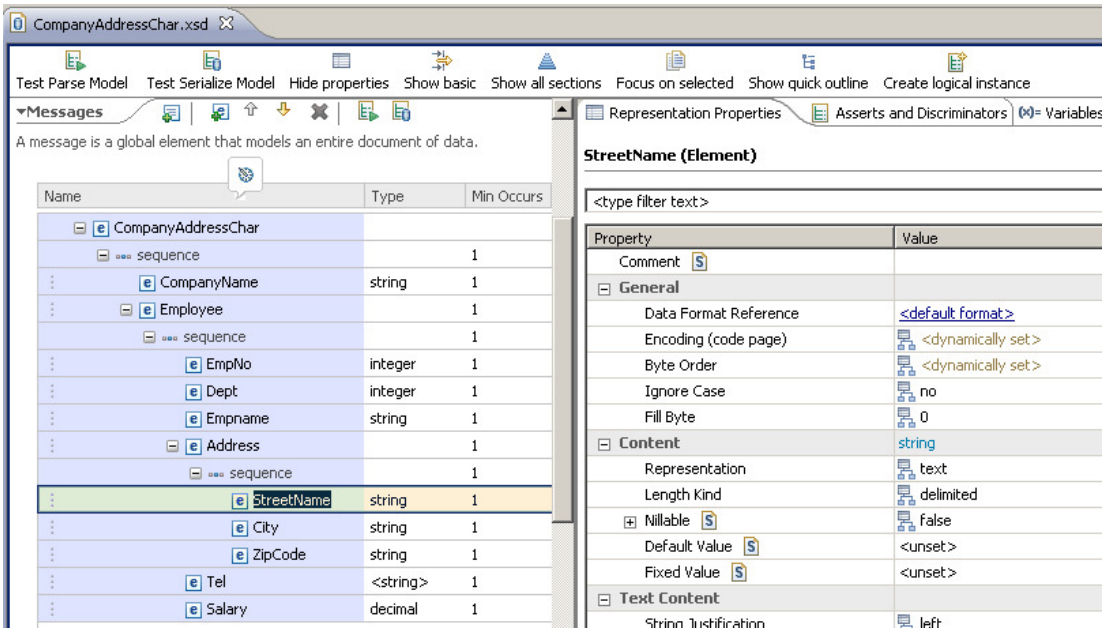
11. You have now defined the Simple Type (TwoCharsText) that we will reference from the elements in the main model.

    Save the model (Ctrl-S).

Provided by IBM BetaWorks

12. Now switch to the CompanyAddressChar message.

    The three elements under the Address element need to be changed to use the TwoCharsText simple type that you just defined.

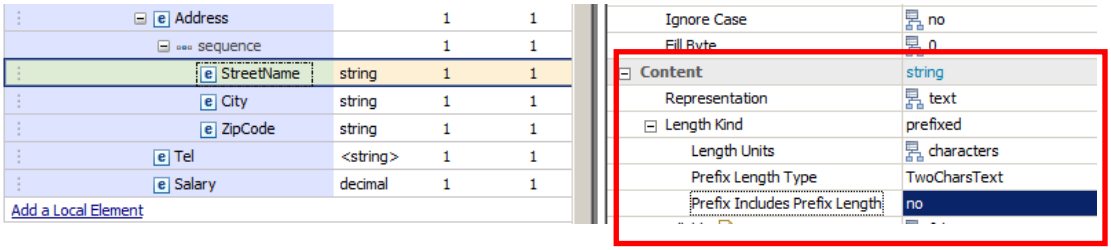


13. Highlight the StreetName element, and make the following changes to the Representation Properties of this element (Content section).

    Representation = text
    Length Kind = prefixed

    When you set the Length Kind to Prefixed, the editor provides further properties which allow you to set additional value. Use the drop-down value to select the following values:

    Length Units = characters
    Prefix Length Type = TwoCharsText
    Prefix Includes Prefix Length = no.



14. Make the same changes to the City and ZipCode elements.

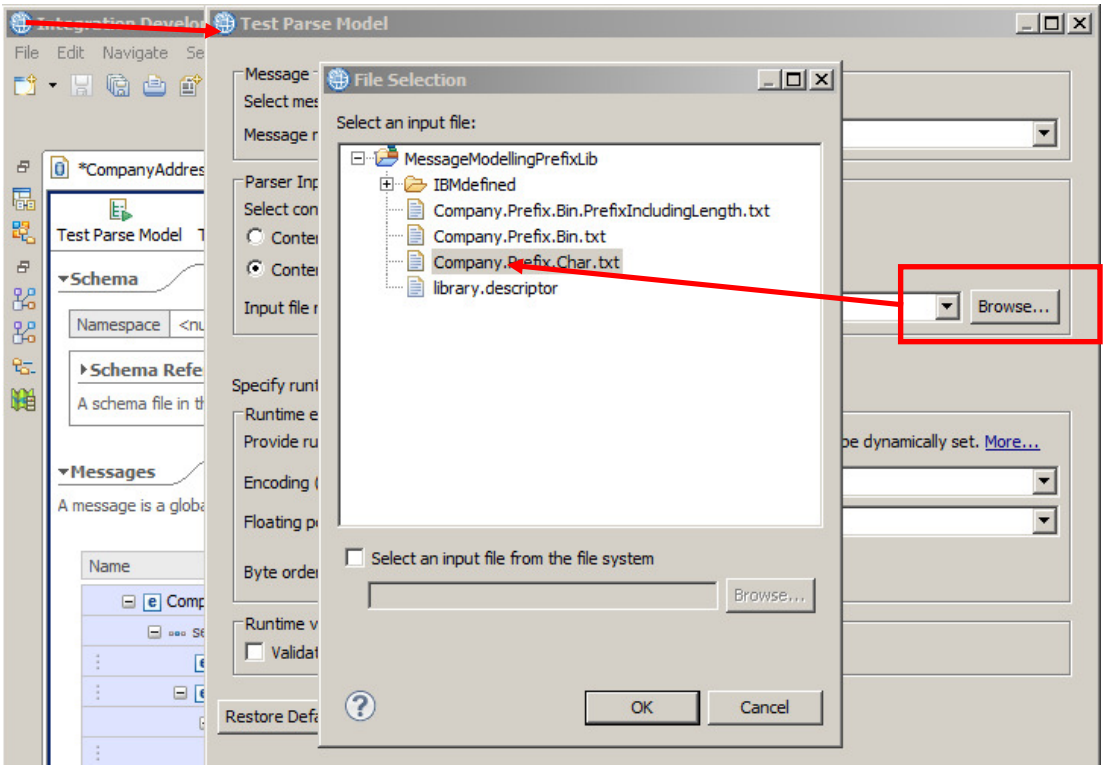
    Save the model.

    Now you are done, and ready to test the new model!

Provided by IBM BetaWorks

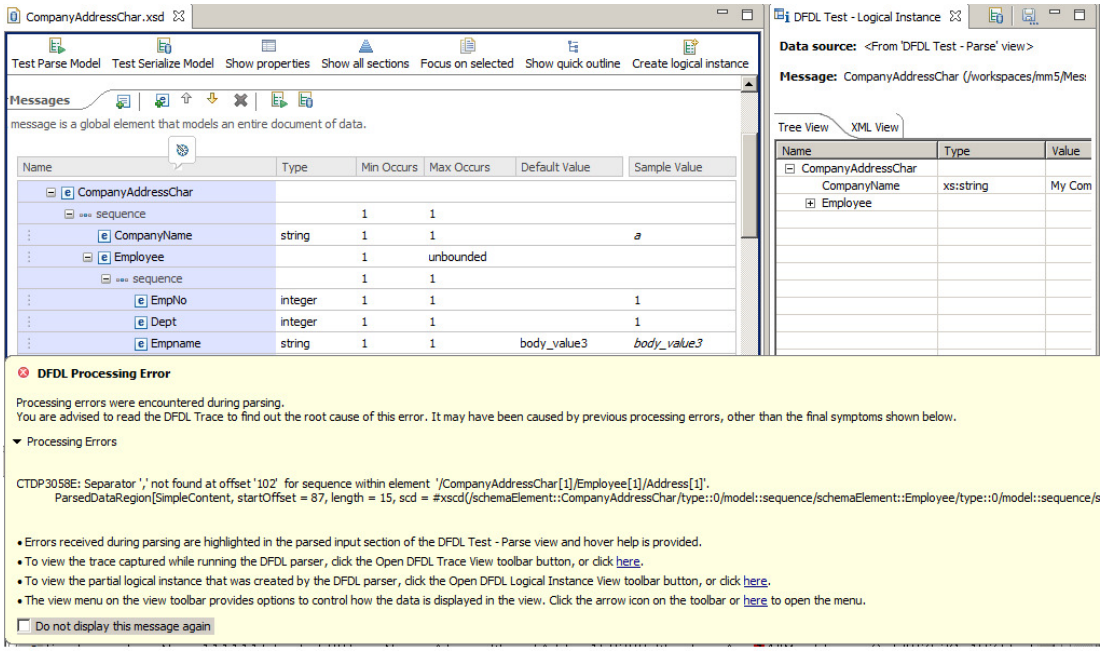
15. Click the Test Parse Model button.

    Select “Content from a data file”, and click Browse. Select the Company.Prefix.Char.txt file, and click OK, and then OK again and then confirm the switch to the DFDL perspective.



16. Success!

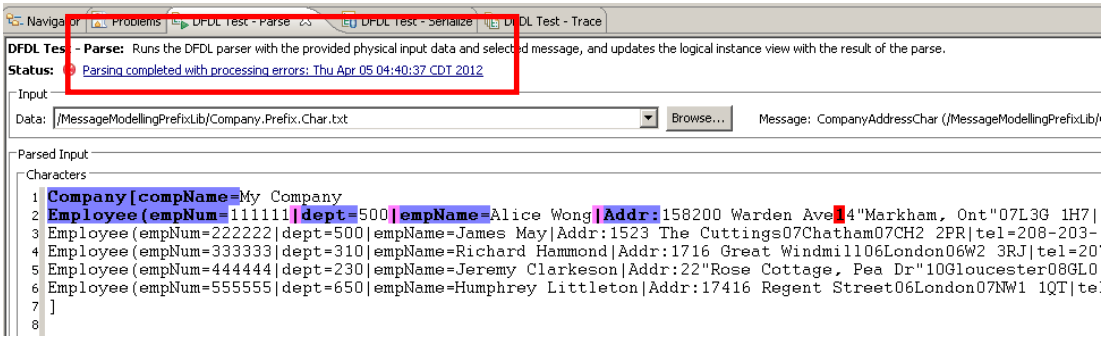
    (Or perhaps not ….)



What did you do wrong?

Close the yellow parser output message.

See if you can work out what went wrong by using the Test Parser output messages, and the highlighting in the Test - Parse window. You may also find it useful to take a look at the parse trace file, easily accessed by clicking on the link in the Test - Parse window.



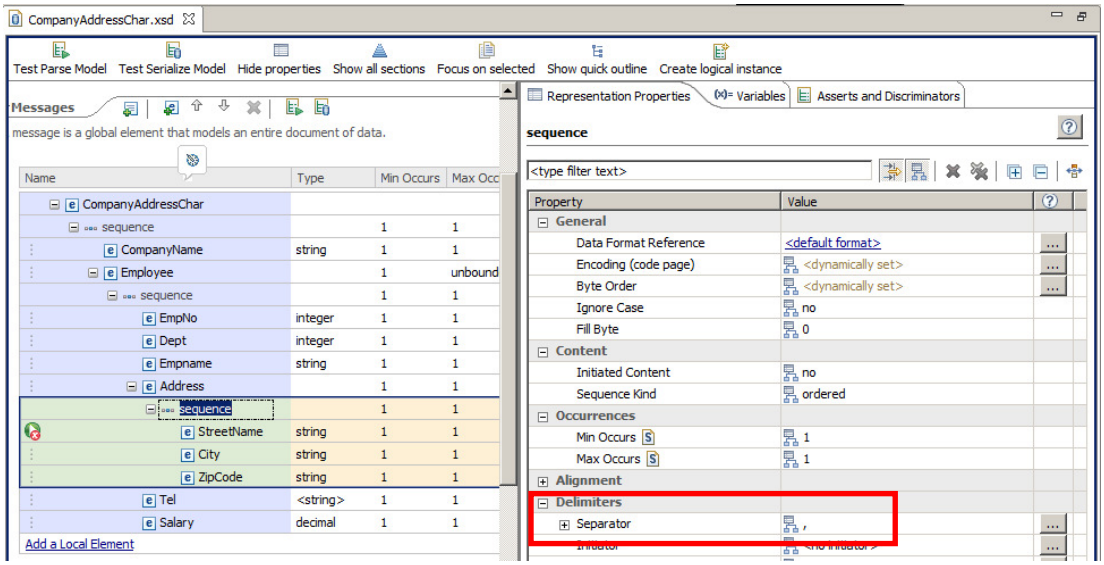
If you can't work this out, proceed to the next page . . .

Page intentionally left blank to give you time to work out what went wrong . . .

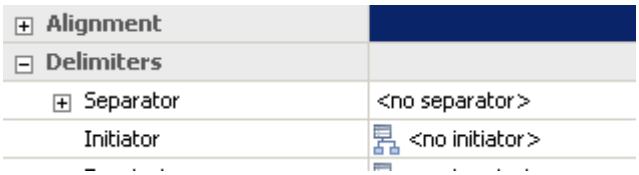
17. Well, the clues are fairly clear in fact. The parse failure message says that a separator is missing for a sequence within the Address element. Now the changes that you have made in this lab have changed the parsing of the elements under Address from using a separator, to using the prefix length. So, why is the model still expecting a separator (and not finding one in the test data).

    Come to think of it, you didn't actually make a change to the separator definition, did you?

    Switch back to the Integration Development perspective, and take a look at the Address sequence field in the editor. You will see that the separator for the sequence element is still set to ',' (ie. a comma). So, the model is expecting these fields to be delimited by a comma, and of course our data does not match this model.



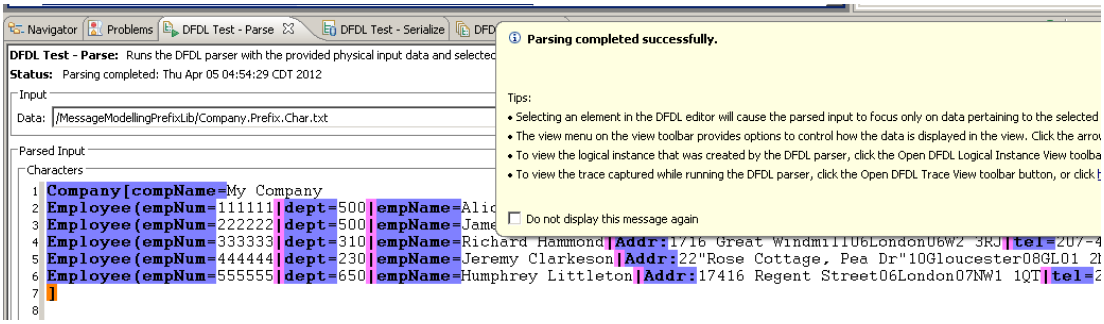
18. Change the separator to "no separator" (use the delete key …. do not set the separator to a blank character).



    Save the model.

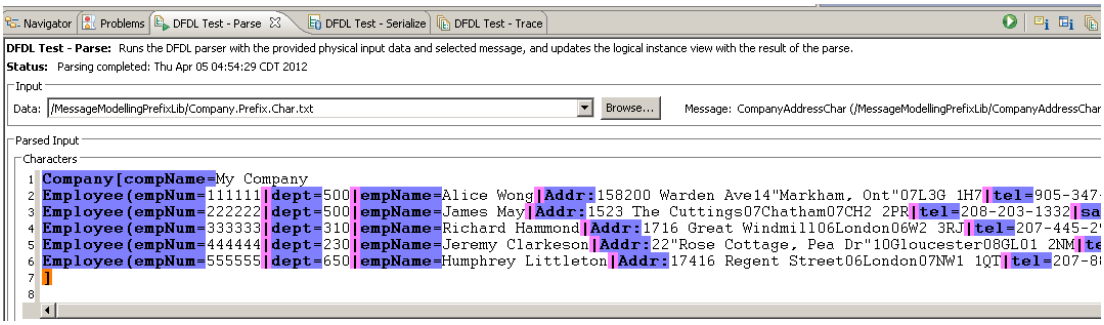
Provided by IBM BetaWorks

19. Now retest the model. This time… success !



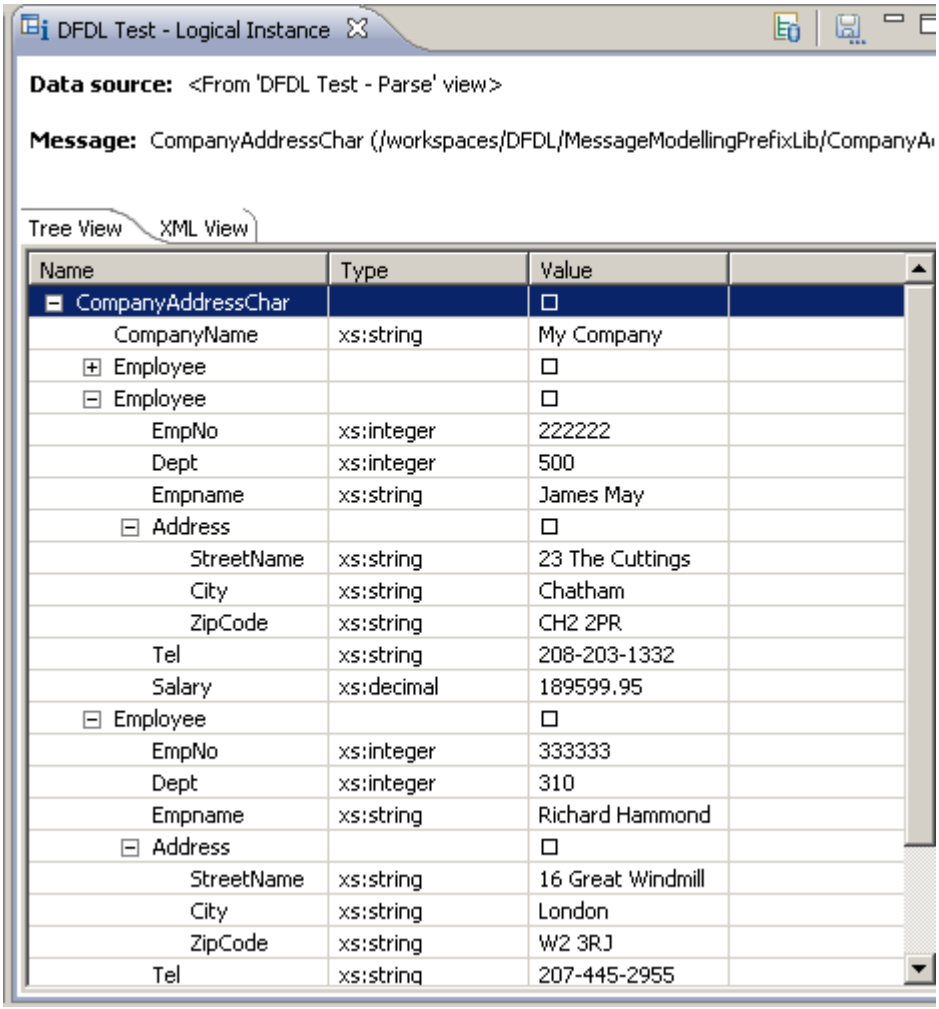
20. Close the yellow completion pop-up.

    The parsed data will be seen in the Test – Parse window.



Provided by IBM BetaWorks

21. In the Logical Instance window, expand the Tree View, and expand the Address element in one or two of the employee elements. You will see that the message has been fully parsed. The prefix length does not show in the Tree View (it is not treated as part of the message data), although it is displayed in the Test - Parse window.



Name	Type	Value
CompanyAddressChar		
CompanyName	xs:string	My Company
Employee		
Employee		
EmpNo	xs:integer	222222
Dept	xs:integer	500
Empname	xs:string	James May
Address		
StreetName	xs:string	23 The Cuttings
City	xs:string	Chatham
ZipCode	xs:string	CH2 2PR
Tel	xs:string	208-203-1332
Salary	xs:decimal	189599.95
Employee		
EmpNo	xs:integer	333333
Dept	xs:integer	310
Empname	xs:string	Richard Hammond
Address		
StreetName	xs:string	16 Great Windmill
City	xs:string	London
ZipCode	xs:string	W2 3RJ
Tel	xs:string	207-445-2955

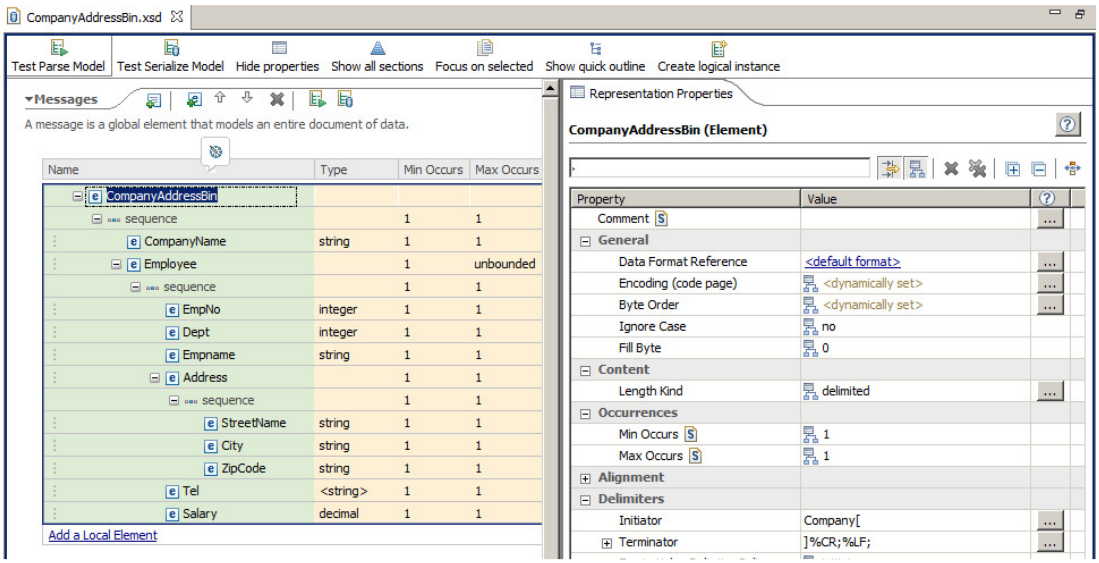
This concludes the Prefix Length Character scenario.

Provided by IBM BetaWorks

4. Create the Prefix Length Binary scenario

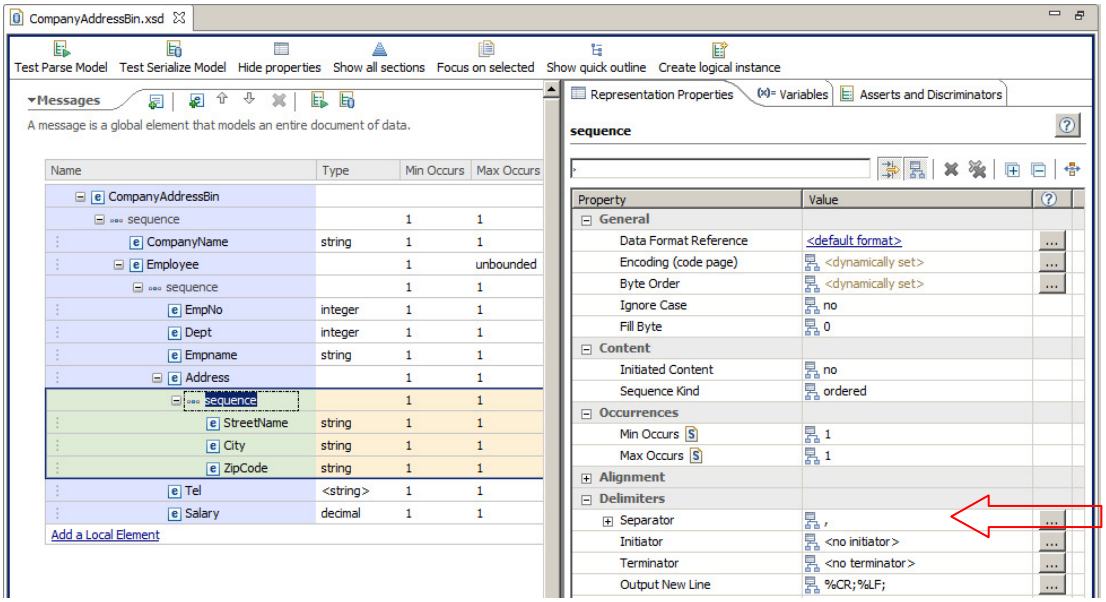
1. Close the Test Parse perspective, and close the CompanyAddressChar message model.

   Open and expand the CompanyAddressBin.xsd message model.



2. Highlight the Address element. You will see that the separator has been set to ‘,’; this means that all fields in the Address element are separated by commas.

   This is the part of the model that we will change.



3. You will change the elements within the Address element to be identified and parsed by using length prefixes.

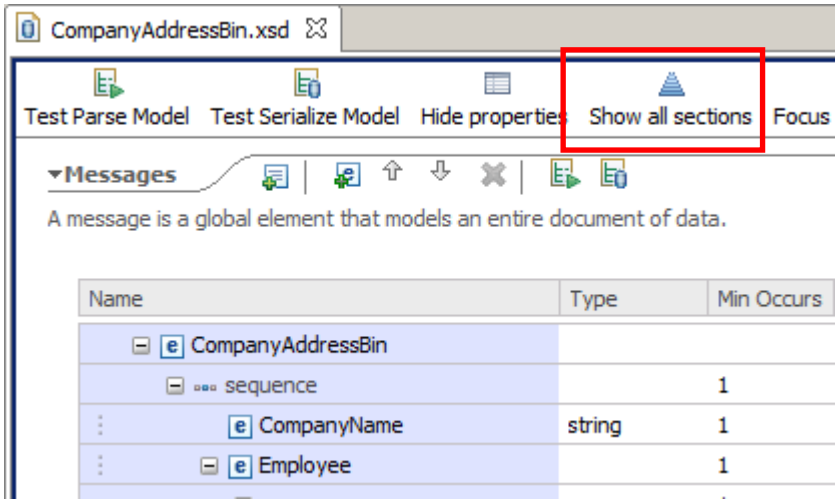
   In this scenario, each of the elements under the Address element will have a prefix of length 2 bytes. The prefix will indicate the length of the element, and the value of the prefix will be a two's complement binary integer. In this case, the value contained in the length prefix will include the length of the prefix itself, unlike the character scenario.

   The Address element may look like this:

   ```
   Addr: ¤8200 Warden Ave "Markham, Ont" L3G 1H7
   ```

4. To define this type of model, you first need to define a Simple Type. This is used to define the physical characteristics of the prefix length. An element which has a prefix length then simply refers to the simple type.

   Click "Show all sections" on the main editor line.

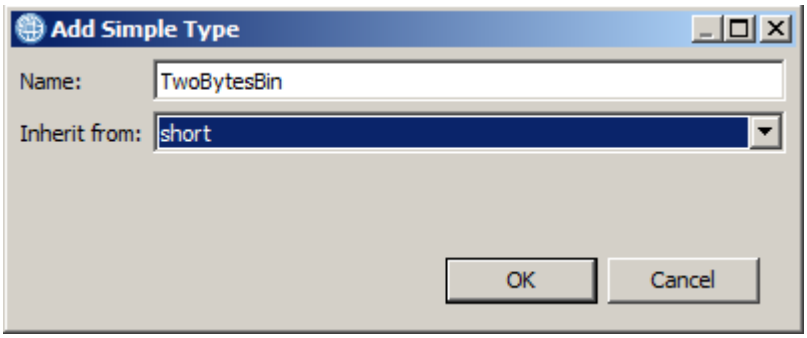


Provided by IBM BetaWorks

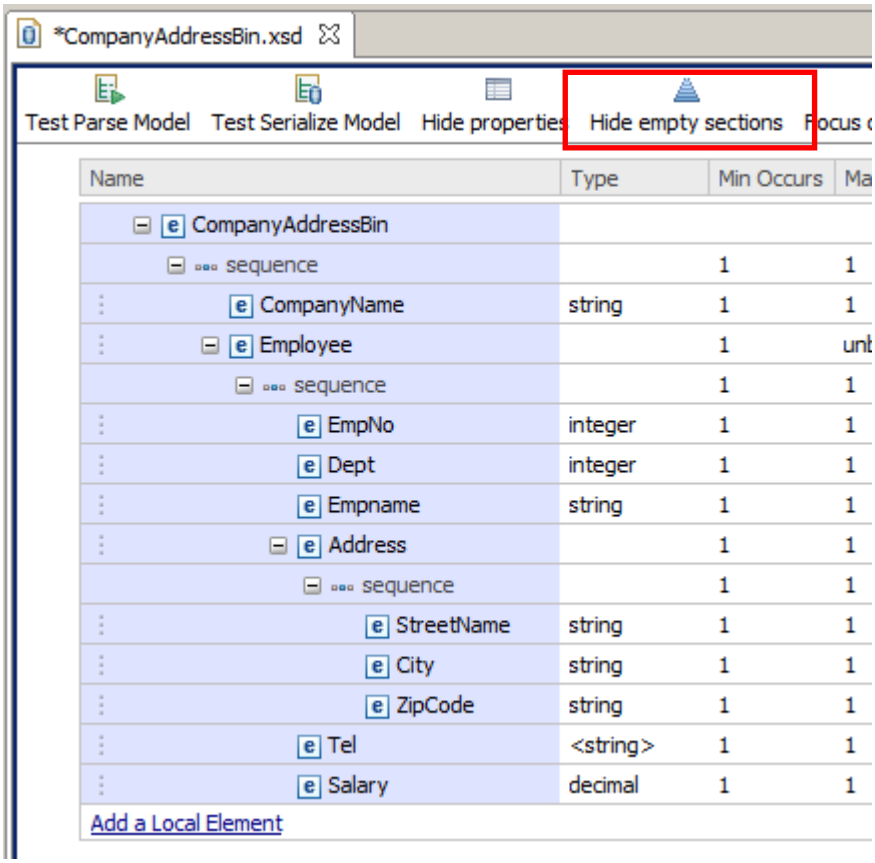
5. In the main editor pane, expand Simple Types, and then click the “Add Simple Type” button.

6. In the dialogue window, set Name = TwoBytesBin (you can define your own descriptive name for this type), and set “Inherit from” to short.

   Click OK.

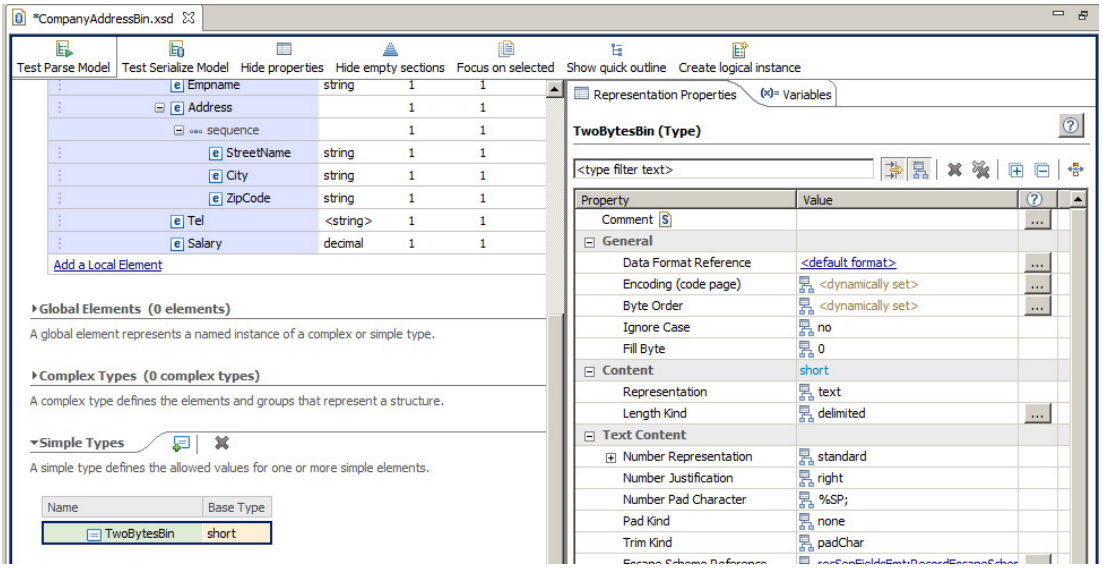


7. To make the editor clearer, click “Hide empty sections”.



Provided by IBM BetaWorks

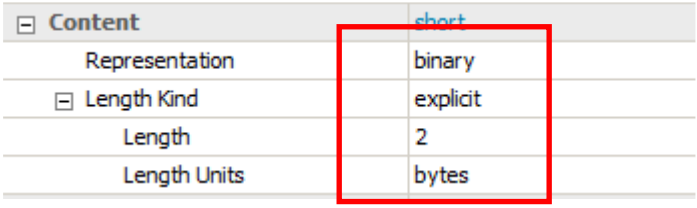
8. Highlight the new Simple Type. You will see that various properties have been set for this new type, shown in the Representation Properties in the right hand pane. Some of these properties must be changed to reflect the nature of our prefix length values.



9. In the Content section, Representation has been set to “text”. Change this to “binary”.

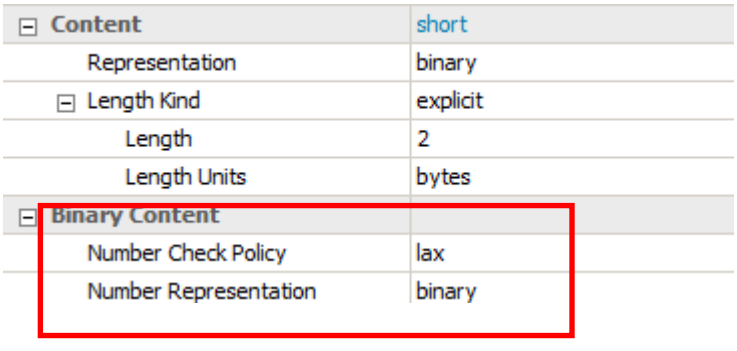
   Second, the “Length kind” has been set to “delimited”. Change this to “explicit”.

   The editor will then provide two further properties. Set Length to 2, and set Length Units to “bytes”.



10. Finally, when the number representation is “binary", the “Binary Number Representation” must have a defined value. Set this to “binary”. This means that the value is a “two's complement” integer.

    Set the Binary Number Check Policy to “lax”. It is required to set this property for binary elements, but you can also set it to “Strict” in this example.



Provided by IBM BetaWorks

11. You have now defined the Simple Type (TwoBytesBin) that we will reference from the elements in the main model.

    Save the model (Ctrl-S).

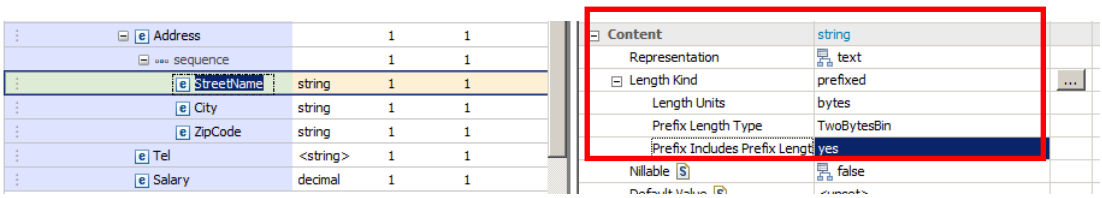
12. Now switch to the CompanyAddressBin model.

    The three elements under the Address element need to be changed to use the TwoBytesBin simple type element that you just defined.

13. Highlight the StreetName element, and make the following changes to the Representation Properties of this element (Content section).

    Representation = text
    Length Kind = prefixed

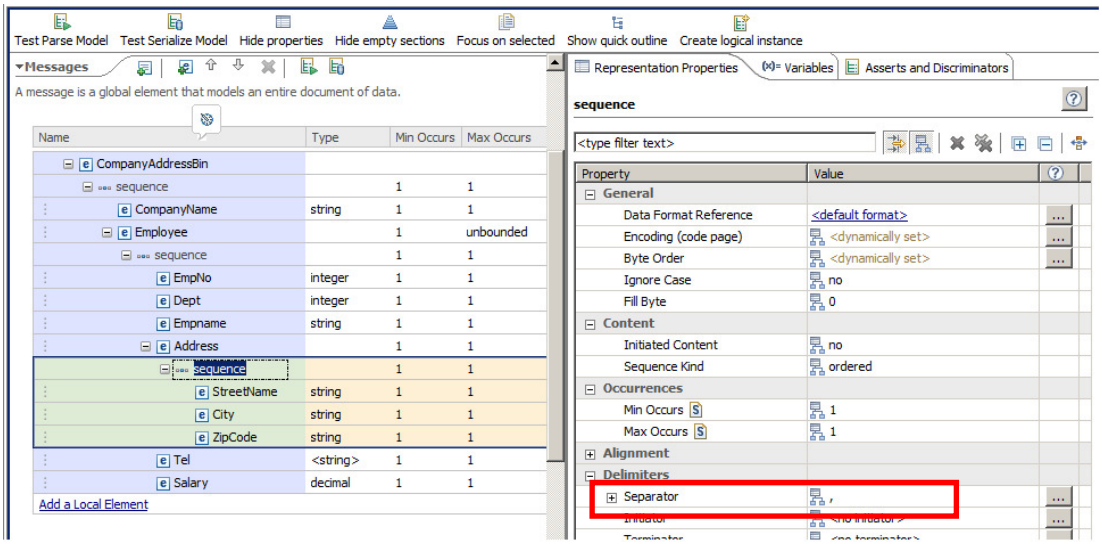
    When Length Kind is set to “prefixed”, further properties should be set as follows:
    Length Units = bytes
    Prefix Length Type = TwoBytesBin
    Prefix Includes Prefix Length = yes (this means the length value will include the length of the prefix itself)



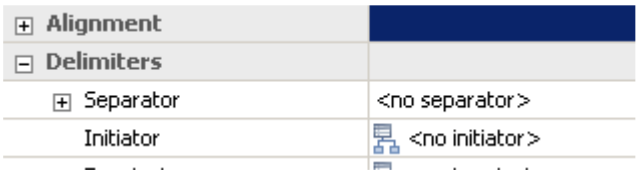
    Make the same changes to the City and ZipCode elements.

14. As in the first scenario, you now need to remove the separator from the Address sequence.

    You will see that the separator for the sequence element is still set to ‘,’ (ie. a comma).



15. Change the separator to “no separator” (use the delete key …. do not set the separator to a blank character).



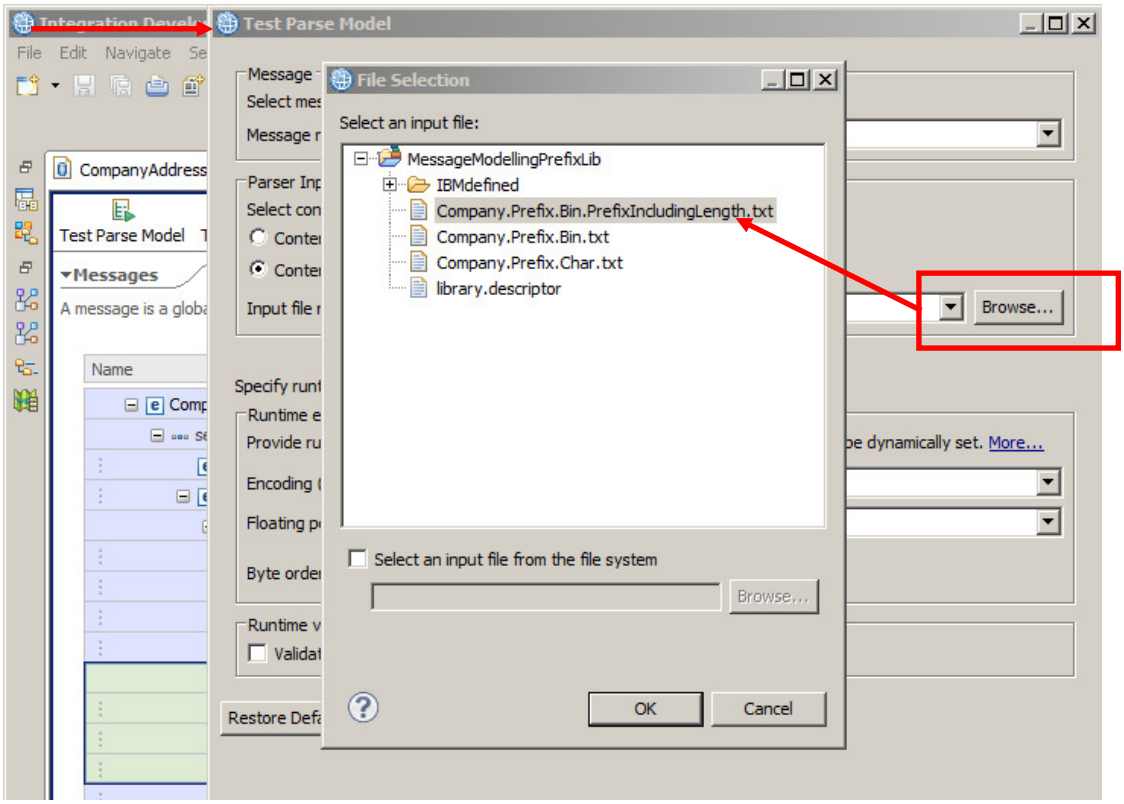
    Save the model.

Provided by IBM BetaWorks

16. Click the Test Parse Model button.

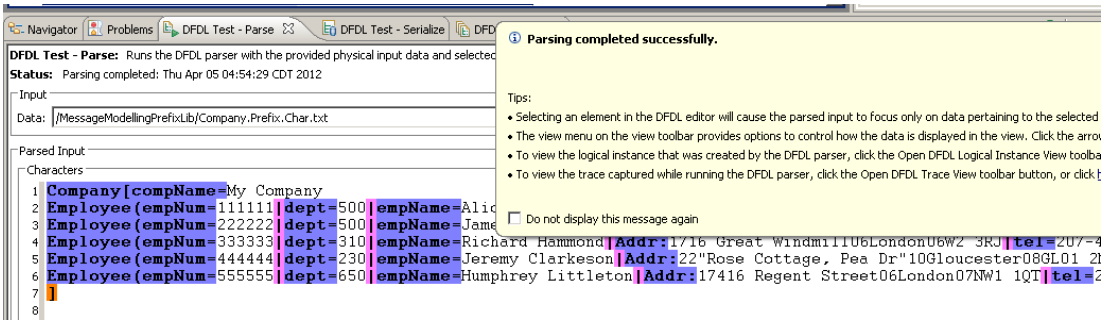
    Select “Content from a data file”, and click Browse. Select the Company.Prefix.Bin.PrefixIncludingLength.txt file, and click OK, and then OK again.

    Do not use the file Company.Prefix.Bin.txt …. that is a test file with data where the length prefix does not contain the length of the prefix itself….. that model is left as an exercise for the reader.

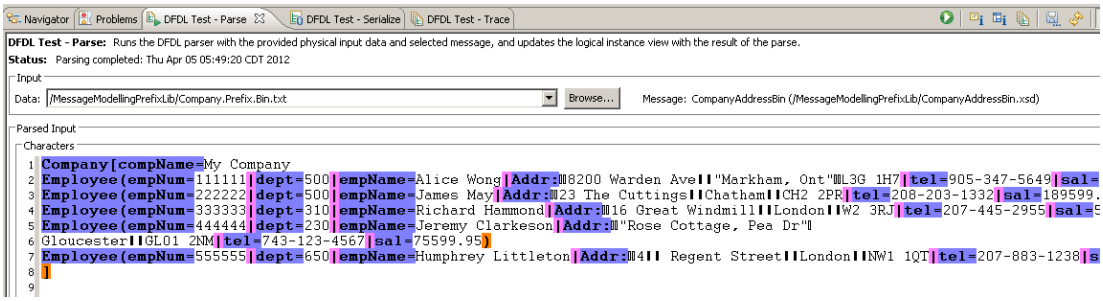


Provided by IBM BetaWorks

17. Close the yellow completion pop-up.

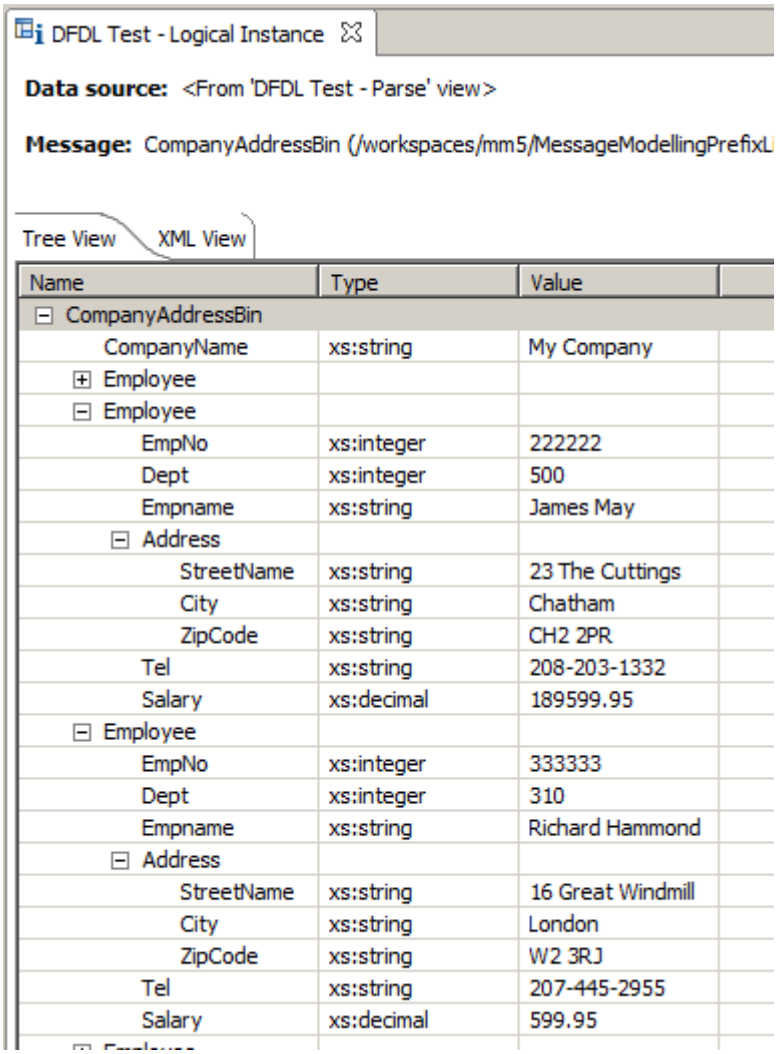


The parsed data will be seen in the Test – Parse window.



Provided by IBM BetaWorks

18. In the Logical Instance window, expand the Tree View, and expand the Address element in one or two of the employee elements. You will see that the message has been fully parsed. The prefix length does not show in the Tree View (it is not treated as part of the message data), although it is displayed in the Test - Parse window.



Name	Type	Value
CompanyAddressBin		
CompanyName	xs:string	My Company
Employee		
Employee		
EmpNo	xs:integer	222222
Dept	xs:integer	500
Empname	xs:string	James May
Address		
StreetName	xs:string	23 The Cuttings
City	xs:string	Chatham
ZipCode	xs:string	CH2 2PR
Tel	xs:string	208-203-1332
Salary	xs:decimal	189599.95
Employee		
EmpNo	xs:integer	333333
Dept	xs:integer	310
Empname	xs:string	Richard Hammond
Address		
StreetName	xs:string	16 Great Windmill
City	xs:string	London
ZipCode	xs:string	W2 3RJ
Tel	xs:string	207-445-2955
Salary	xs:decimal	599.95

END OF LAB GUIDE