IBM Integration Bus

Message Modeling with DFDL

Lab 5
Using DFDL length prefixes

June, 2013

Hands-on lab built at product
code level Version 9.0.0.0

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 ... 22

1. Introduction

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

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:\student directory. This will create a subdirectory called MessageModelling, with several further subdirectories. If you are using the pre-supplied vmware image, this will already be available.

1.2 Lab Scenario

This lab 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”.

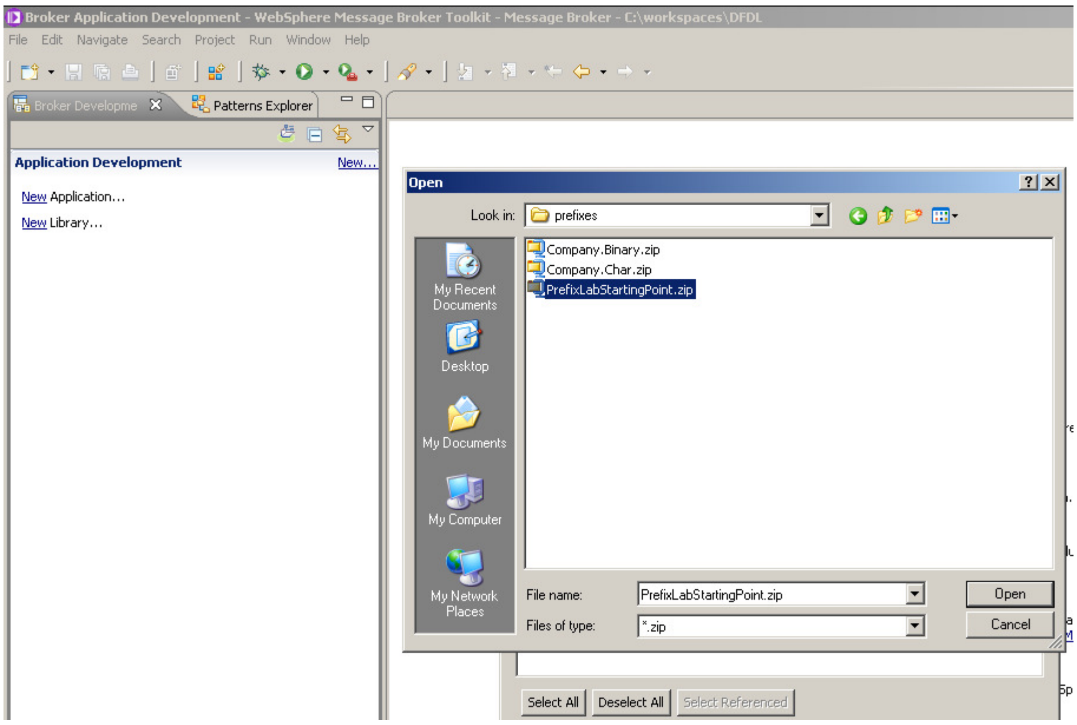
CompanyAddressBin.xsd – some of the elements will be changed to use a 2-byte length prefix of type “binary”.

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:\student\MessageModelling\prefixes\PrefixLabStartingPoint.zip.


Broker Application Development - WebSphere Message Broker Toolkit - Message Broker - C:\workspaces\DFDL

Open

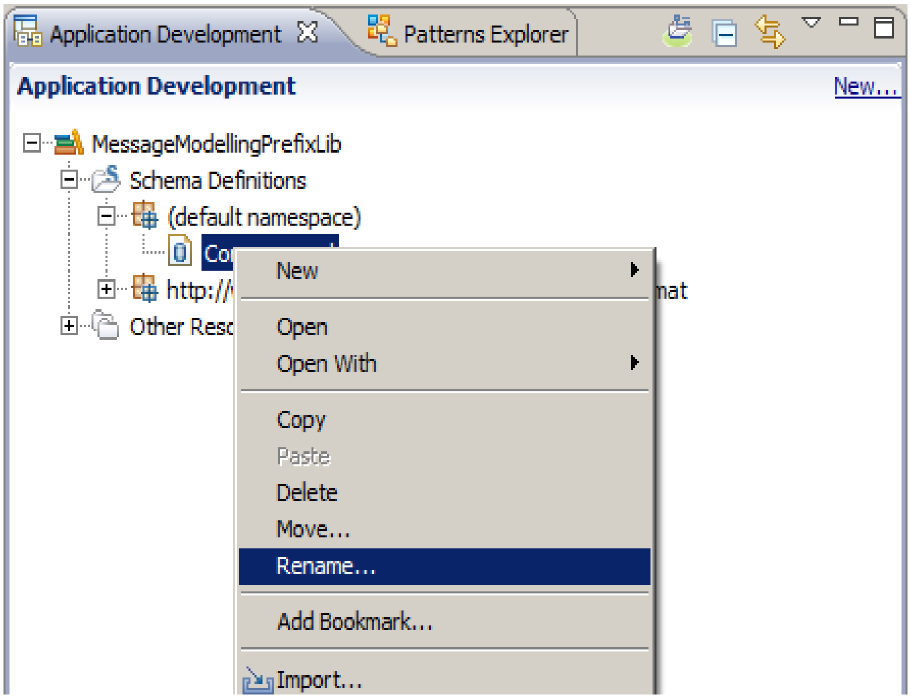
Look in: prefixes

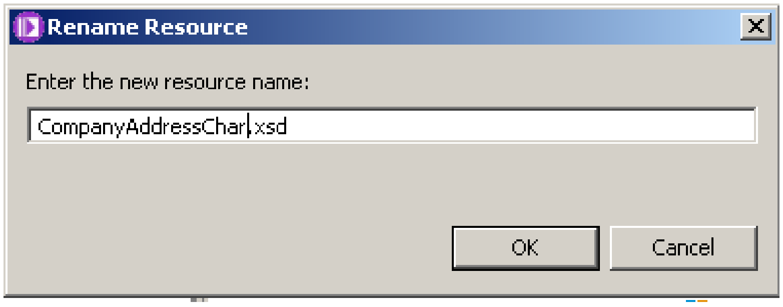
Company.Binary.zip
Company.Char.zip
PrefixLabStartingPoint.zip

File name: PrefixLabStartingPoint.zip

Files of type: *.zip

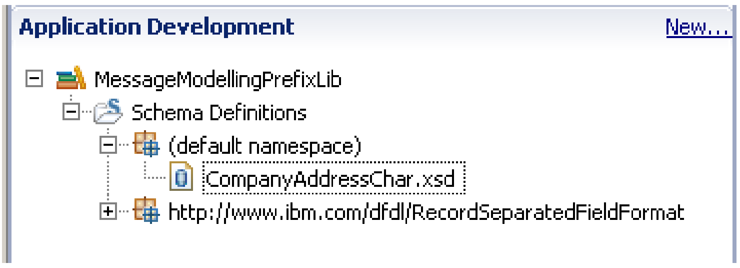
2. Rename the schema Company.xsd to CompanyAddressChar.xsd.




Rename Resource

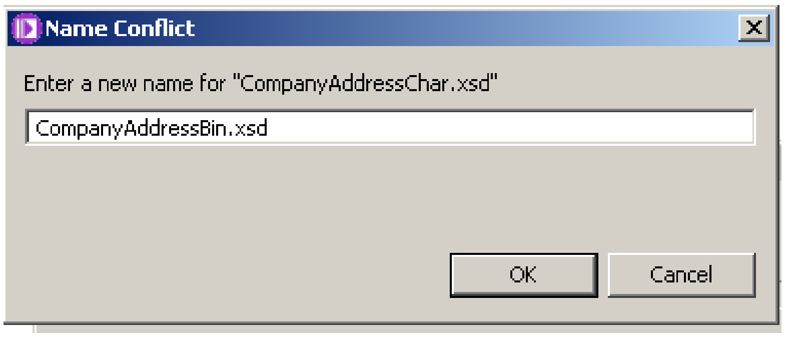
Enter the new resource name:

CompanyAddressChar.xsd



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

   Use Ctrl-C / Ctrl-V.

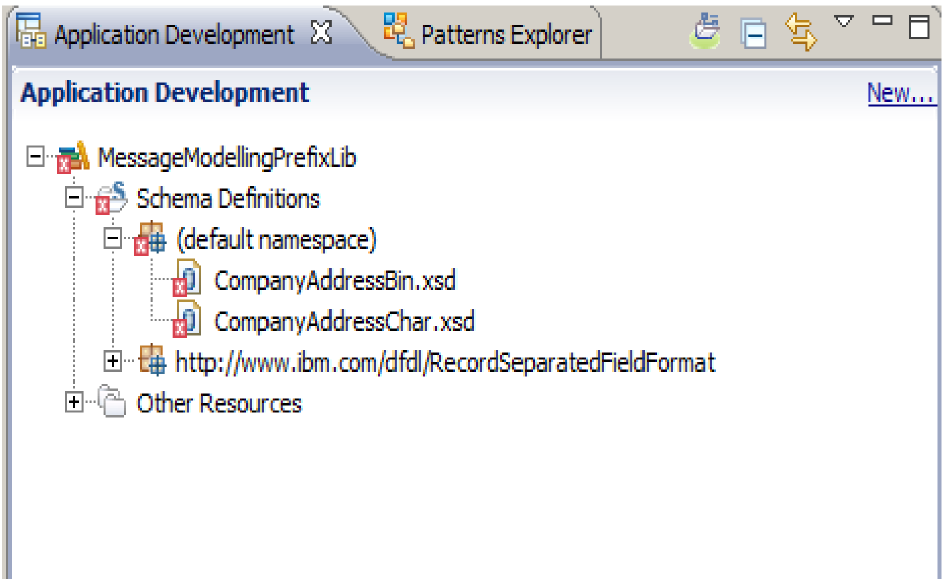


Name Conflict

Enter a new name for "CompanyAddressChar.xsd"

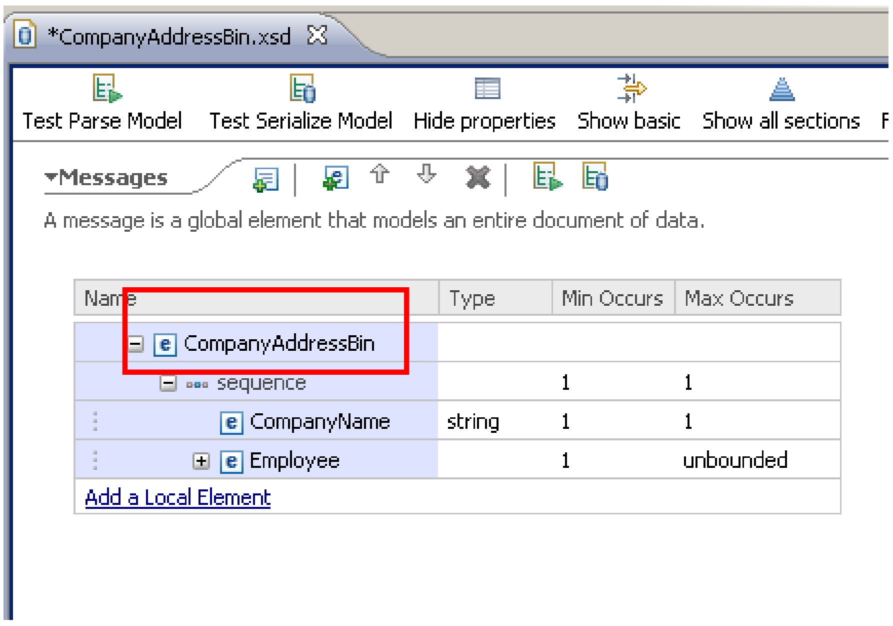
CompanyAddressBin.xsd

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.



```
MessageModellingPrefixLib
  Schema Definitions
    (default namespace)
      CompanyAddressBin.xsd
      CompanyAddressChar.xsd
    http://www.ibm.com/dfdl/RecordSeparatedFieldFormat
  Other Resources
```

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



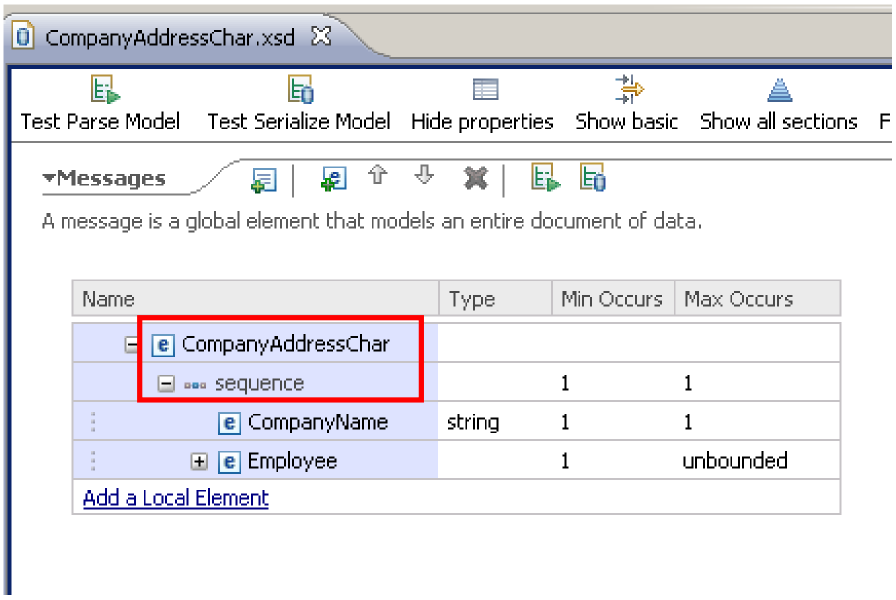
*CompanyAddressBin.xsd

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

Name	Type	Min Occurs	Max Occurs
CompanyAddressBin			
sequence		1	1
CompanyName	string	1	1
Employee		1	unbounded
Add a Local Element			

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.



CompanyAddressChar.xsd

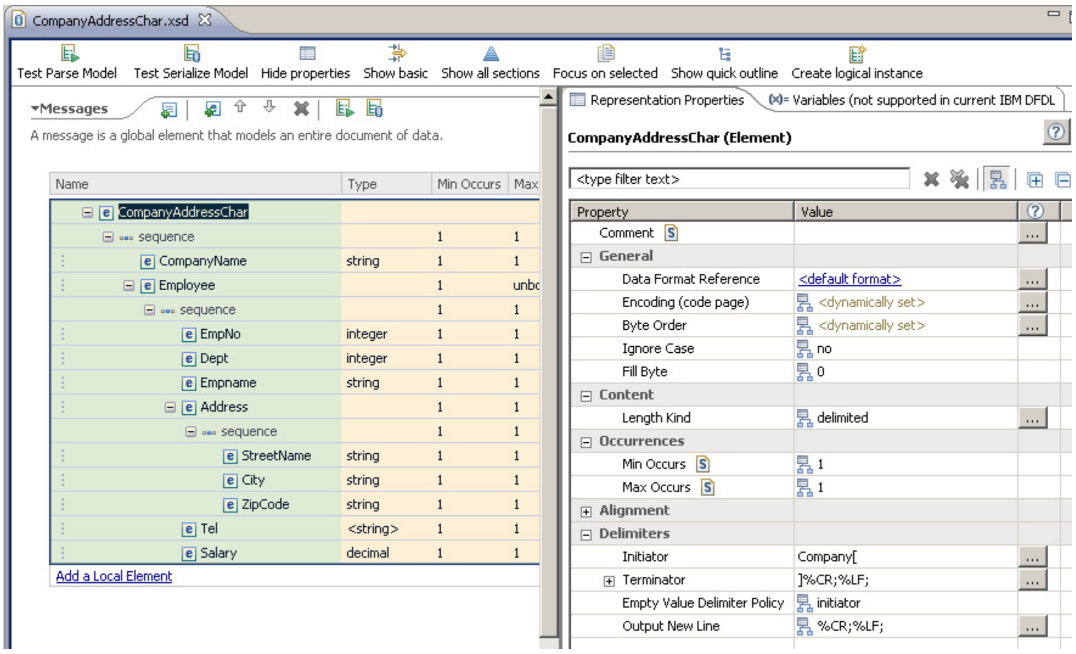
A message is a global element that models an entire document of data.

Name	Type	Min Occurs	Max Occurs
CompanyAddressChar			
sequence		1	1
CompanyName	string	1	1
Employee		1	unbounded
Add a Local Element			

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

3. Create the Prefix Length Character scenario

1. Open and expand the CompanyAddressChar.xsd message model.



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

Name	Type	Min Occurs	Max
CompanyAddressChar			
sequence		1	1
CompanyName	string	1	1
Employee		1	unbc
sequence		1	1
EmpNo	integer	1	1
Dept	integer	1	1
Empname	string	1	1
Address		1	1
sequence		1	1
StreetName	string	1	1
City	string	1	1
ZipCode	string	1	1
Tel	<string>	1	1
Salary	decimal	1	1

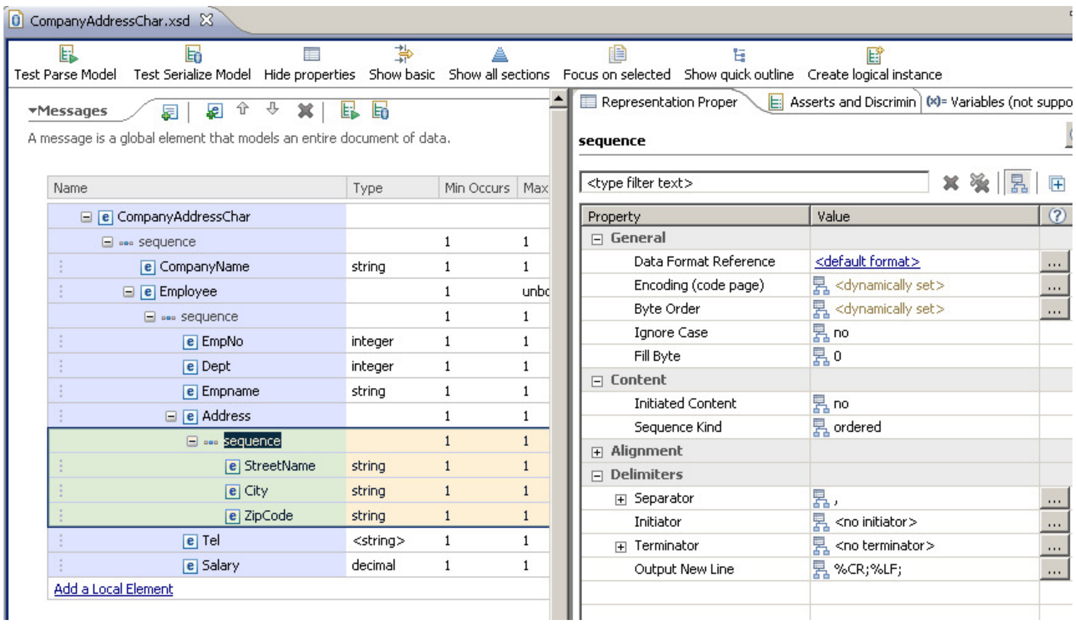
CompanyAddressChar (Element)

<type filter text>

Property	Value
Comment	
General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	no
Fill Byte	0
Content	
Length Kind	delimited
Occurrences	
Min Occurs	1
Max Occurs	1
Alignment	
Delimiters	
Initiator	Company[
Terminator	]%CR;%LF;
Empty Value Delimiter Policy	initiator
Output New Line	%CR;%LF;

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.



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

Name	Type	Min Occurs	Max
CompanyAddressChar			
sequence		1	1
CompanyName	string	1	1
Employee		1	unbo
sequence		1	1
EmpNo	integer	1	1
Dept	integer	1	1
Empname	string	1	1
Address		1	1
sequence		1	1
StreetName	string	1	1
City	string	1	1
ZipCode	string	1	1
Tel	<string>	1	1
Salary	decimal	1	1

sequence

Property	Value
General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	no
Fill Byte	0
Content	
Initiated Content	no
Sequence Kind	ordered
Alignment	
Delimiters	
Separator	,
Initiator	<no initiator>
Terminator	<no terminator>
Output New Line	%CR;%LF;

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:

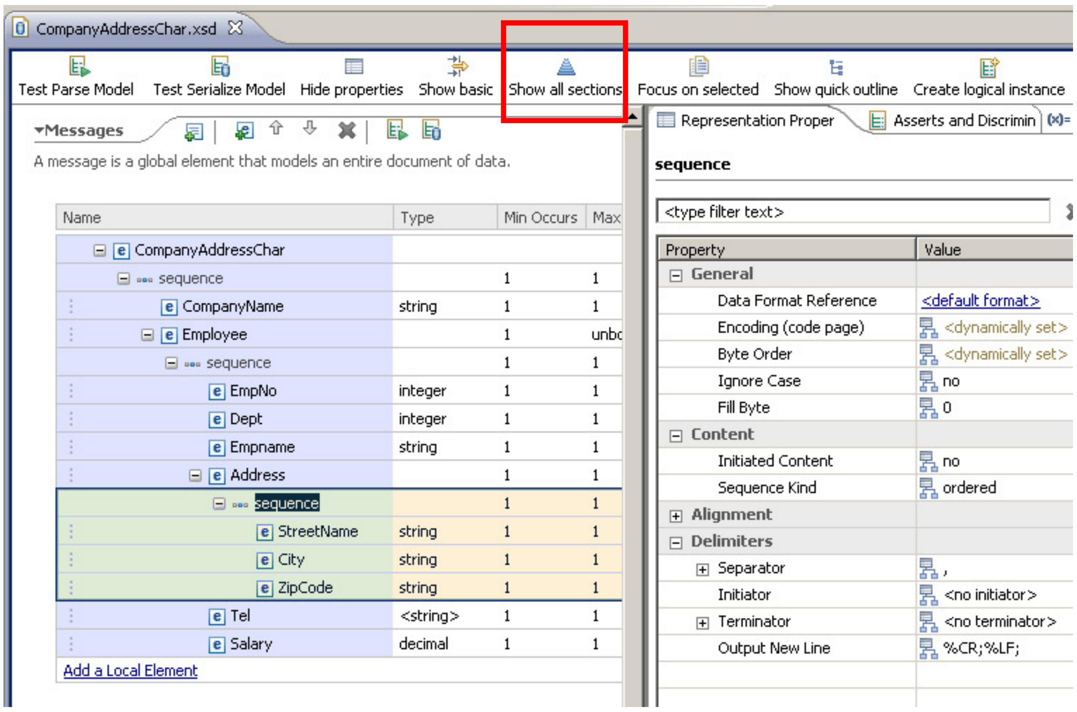
   ```
   Addr:158200 Warden Ave14"Markham, Ont"07L3G 1H7
   ```

   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.



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

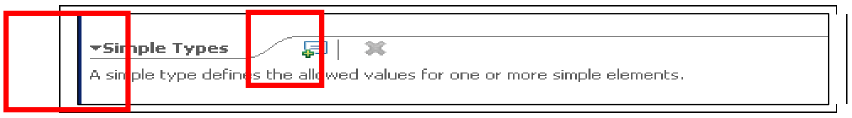
Name	Type	Min Occurs	Max
CompanyAddressChar			
sequence		1	1
CompanyName	string	1	1
Employee		1	unbc
sequence		1	1
EmpNo	integer	1	1
Dept	integer	1	1
Empname	string	1	1
Address		1	1
sequence		1	1
StreetName	string	1	1
City	string	1	1
ZipCode	string	1	1
Tel	<string>	1	1
Salary	decimal	1	1

Add a Local Element

sequence

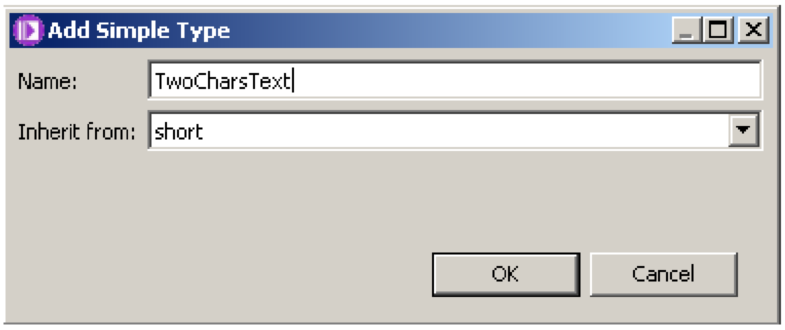
Property	Value
General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	no
Fill Byte	0
Content	
Initiated Content	no
Sequence Kind	ordered
Alignment	
Delimiters	
Separator	,
Initiator	<no initiator>
Terminator	<no terminator>
Output New Line	%CR;%LF;

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



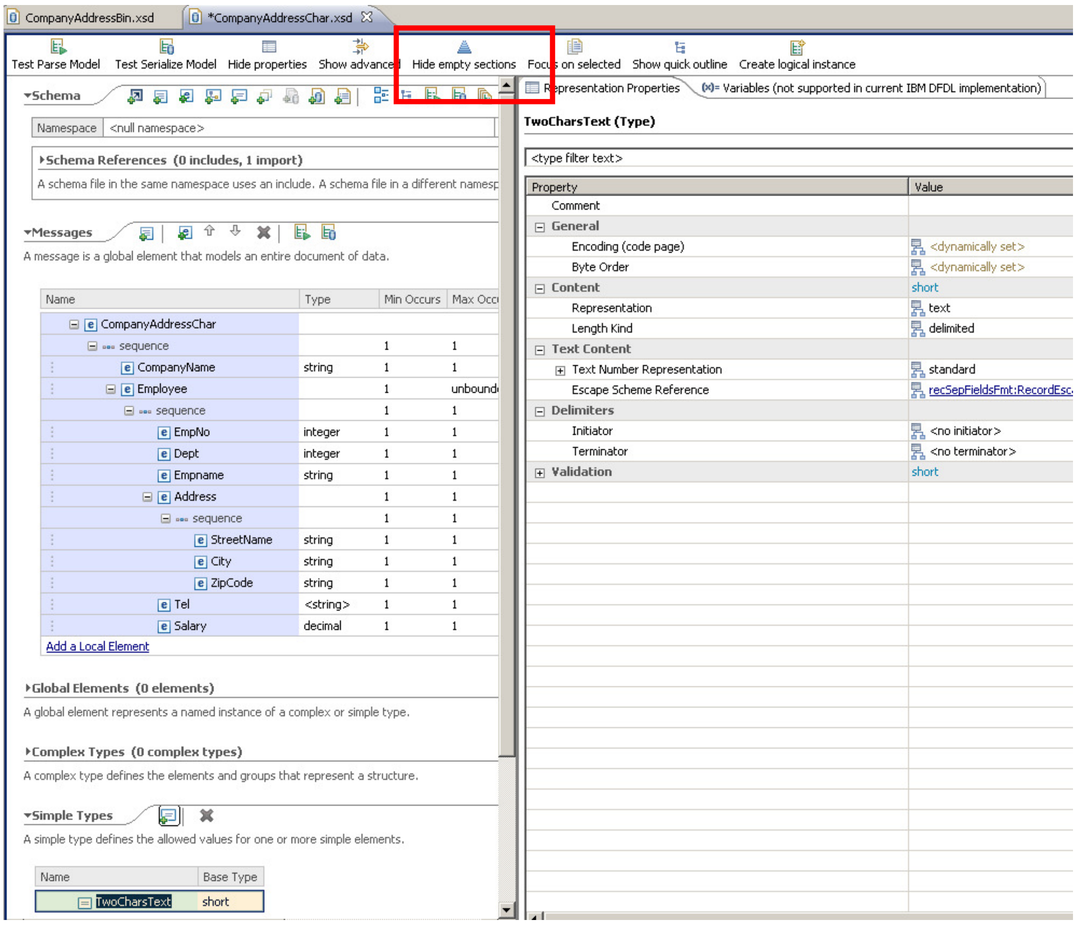
A simple type defines the allowed values for one or more simple elements.

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.



Name:	TwoCharsText
Inherit from:	short

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



Namespace <null namespace>

Schema References (0 includes, 1 import)

A schema file in the same namespace uses an include. A schema file in a different namesp

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

Name	Type	Min Occurs	Max Occu
CompanyAddressChar			
sequence		1	1
CompanyName	string	1	1
Employee		1	unbounde
sequence		1	1
EmpNo	integer	1	1
Dept	integer	1	1
Empname	string	1	1
Address		1	1
sequence		1	1
StreetName	string	1	1
City	string	1	1
ZipCode	string	1	1
Tel	<string>	1	1
Salary	decimal	1	1

Global Elements (0 elements)

A global element represents a named instance of a complex or simple type.

Complex Types (0 complex types)

A complex type defines the elements and groups that represent a structure.

A simple type defines the allowed values for one or more simple elements.

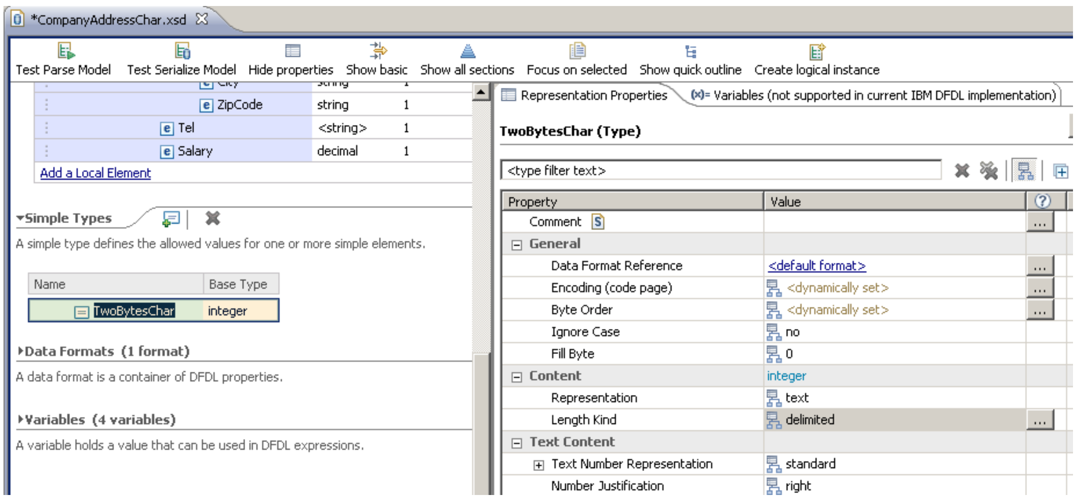
Name	Base Type
TwoCharsText	short

TwoCharsText (Type)

<type filter text>

Property	Value
Comment	
General	
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Content	short
Representation	text
Length Kind	delimited
Text Content	
Text Number Representation	standard
Escape Scheme Reference	recSepFieldsFmt:RecordEsc
Delimiters	
Initiator	<no initiator>
Terminator	<no terminator>
Validation	short

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.



ZipCode	string	1
Tel	<string>	1
Salary	decimal	1

Add a Local Element

Simple Types

A simple type defines the allowed values for one or more simple elements.

Name	Base Type
TwoBytesChar	integer

Data Formats (1 format)

A data format is a container of DFDL properties.

Variables (4 variables)

A variable holds a value that can be used in DFDL expressions.

TwoBytesChar (Type)

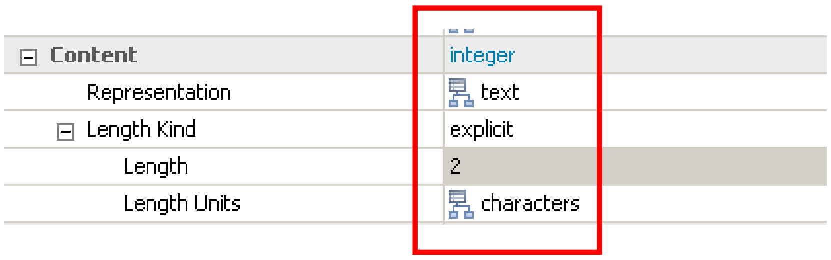
<type filter text>

Property	Value
Comment	
General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	no
Fill Byte	0
Content	integer
Representation	text
Length Kind	delimited
Text Content	
Text Number Representation	standard
Number Justification	right

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”.

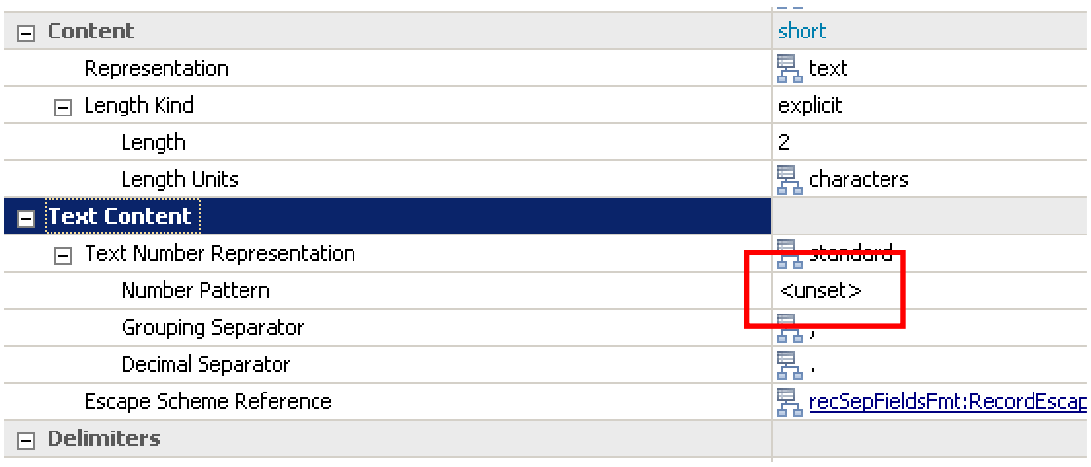
The editor will then provide two further properties. Set Length to 2, and leave Length Units as “characters”.



Content	integer
Representation	text
Length Kind	explicit
Length	2
Length Units	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.

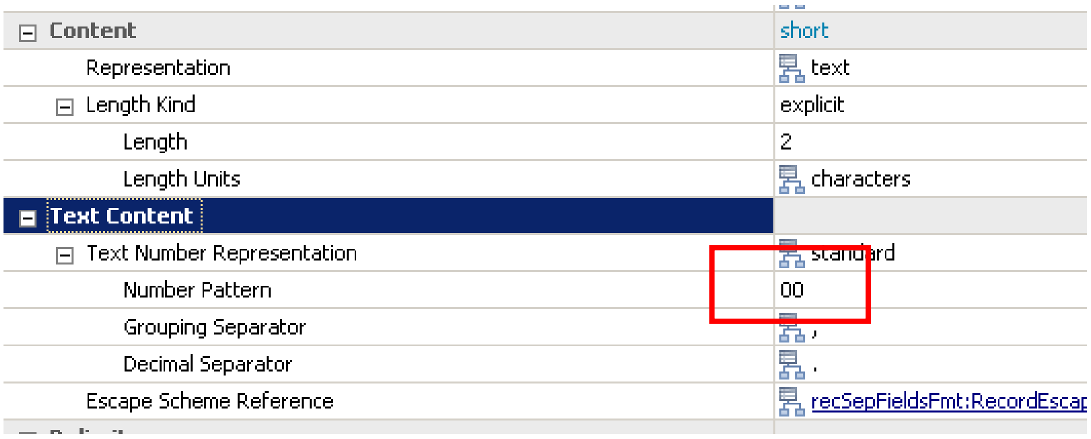
10. Finally, when the number representation is “text”, the “Number Pattern” must have a defined value (it will be set to <unset>).



⊟ Content	short
Representation	text
⊟ Length Kind	explicit
Length	2
Length Units	characters
⊟ Text Content	
⊟ Text Number Representation	standard
Number Pattern	<unset>
Grouping Separator	,
Decimal Separator	.
Escape Scheme Reference	recSepFieldsFmt:RecordEscap
⊟ Delimiters	

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.



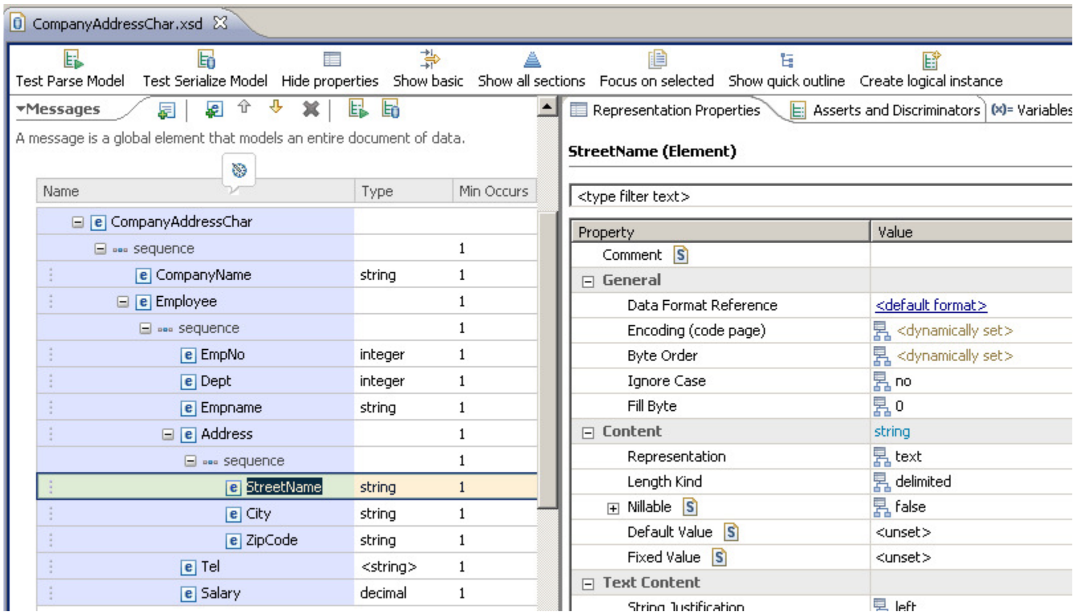
⊟ Content	short
Representation	text
⊟ Length Kind	explicit
Length	2
Length Units	characters
⊟ Text Content	
⊟ Text Number Representation	standard
Number Pattern	00
Grouping Separator	,
Decimal Separator	.
Escape Scheme Reference	recSepFieldsFmt:RecordEscap

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).

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.



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

Name	Type	Min Occurs
CompanyAddressChar		
sequence		1
CompanyName	string	1
Employee		1
sequence		1
EmpNo	integer	1
Dept	integer	1
Empname	string	1
Address		1
sequence		1
StreetName	string	1
City	string	1
ZipCode	string	1
Tel	<string>	1
Salary	decimal	1

StreetName (Element)

<type filter text>

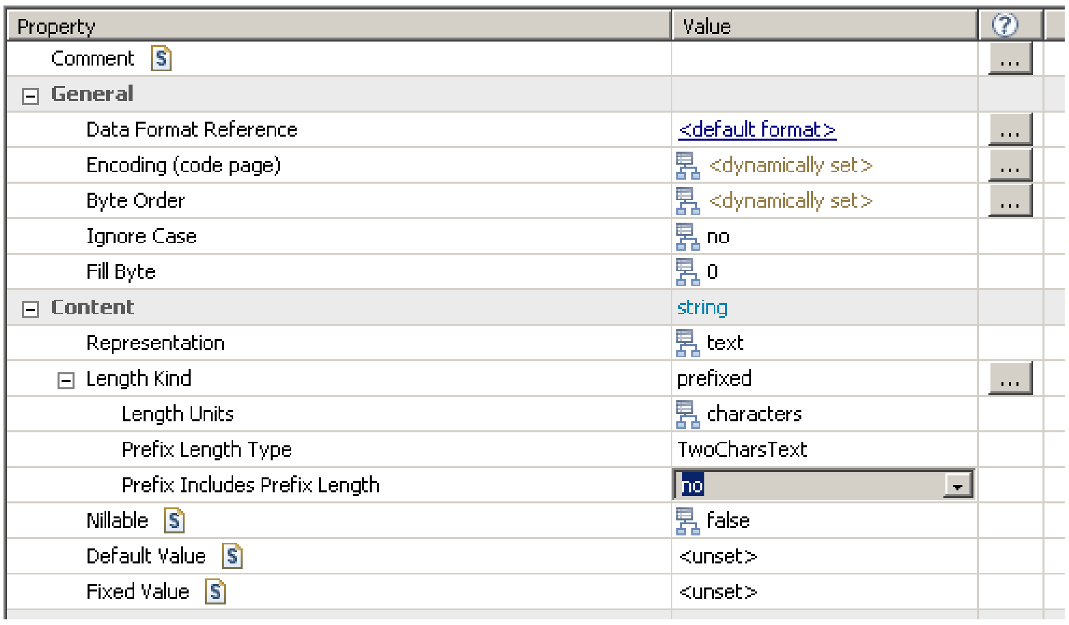
Property	Value
Comment	
General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	no
Fill Byte	0
Content	string
Representation	text
Length Kind	delimited
Nillable	false
Default Value	<unset>
Fixed Value	<unset>
Text Content	
String Justification	left

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.



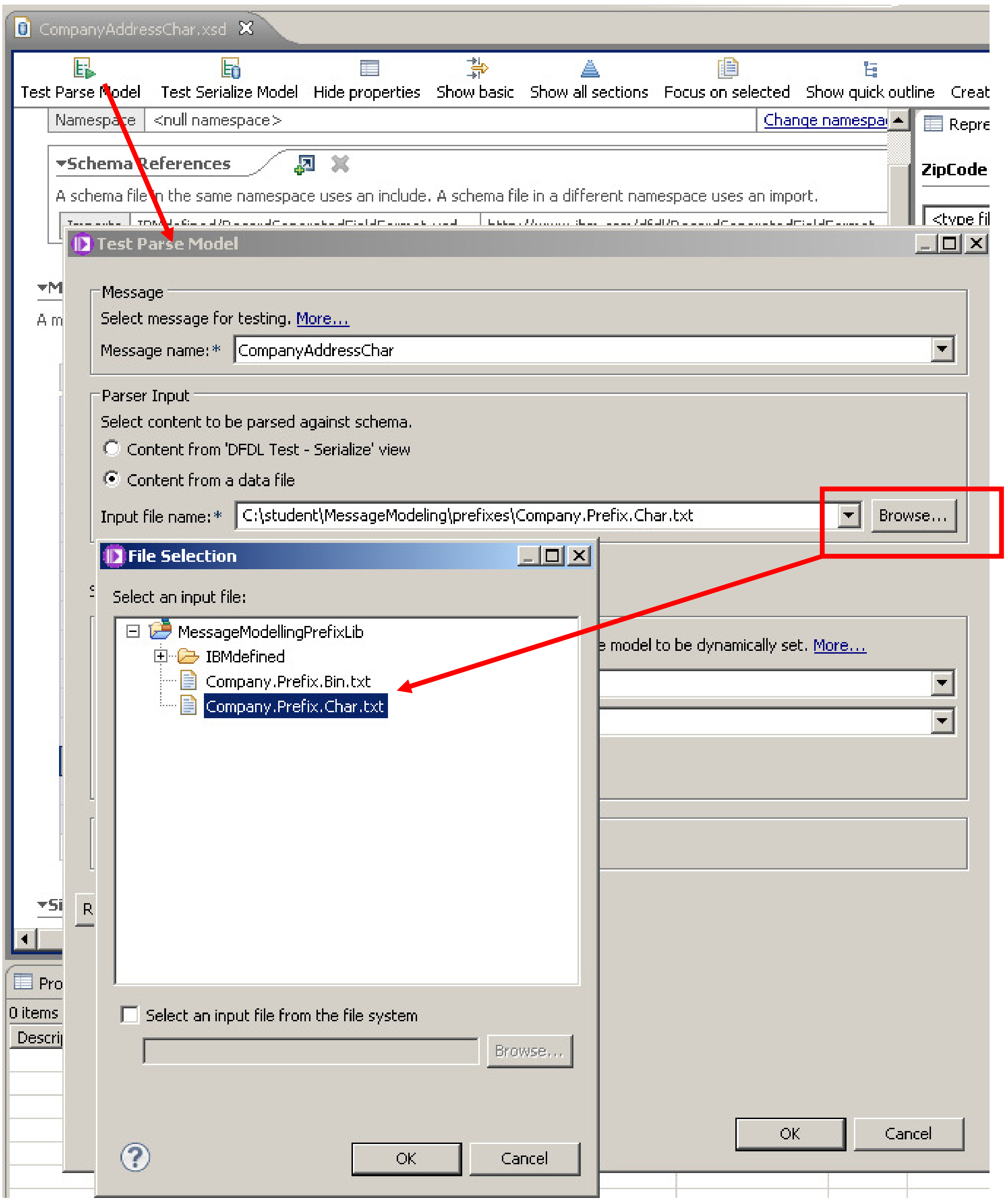
Property	Value
Comment	
⊟ General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	no
Fill Byte	0
⊟ Content	string
Representation	text
⊟ Length Kind	prefixed
Length Units	characters
Prefix Length Type	TwoCharsText
Prefix Includes Prefix Length	no
Nillable	false
Default Value	<unset>
Fixed Value	<unset>

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

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

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.



Message

Select message for testing. More...

Message name:* CompanyAddressChar

Parser Input

Select content to be parsed against schema.

Content from 'DFDL Test - Serialize' view

(●) Content from a data file

Input file name:* C:\student\MessageModeling\prefixes\Company.Prefix.Char.txt Browse...

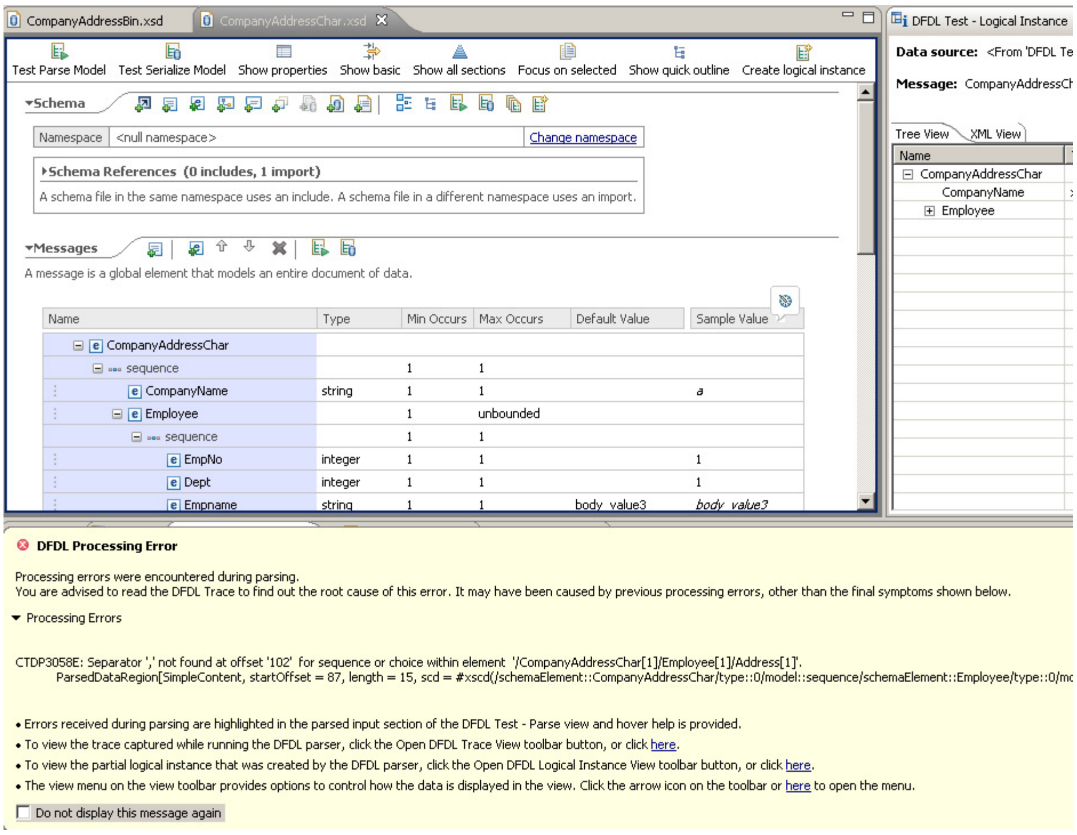
Select an input file:

- MessageModellingPrefixLib
  - IBMdefined
  - Company.Prefix.Bin.txt
  - Company.Prefix.Char.txt

☐ Select an input file from the file system

16. Success !

(Or perhaps not ….)



Schema References (0 includes, 1 import)

A schema file in the same namespace uses an include. A schema file in a different namespace uses an import.

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

Name	Type	Min Occurs	Max Occurs	Default Value	Sample Value
CompanyAddressChar					
sequence		1	1		
CompanyName	string	1	1		a
Employee		1	unbounded		
sequence		1	1		
EmpNo	integer	1	1		1
Dept	integer	1	1		1
Empname	string	1	1	body_value3	*body_value3*

Data source: <From 'DFDL Te

Message: CompanyAddressCh

Name
CompanyAddressChar
CompanyName
Employee

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

```
CTDP3058E: Separator ',' not found at offset '102'  for sequence or choice within element  '/CompanyAddressChar[1]/Employee[1]/Address[1]'.
        ParsedDataRegion[SimpleContent, startOffset = 87, length = 15, scd = #xscd(/schemaElement::CompanyAddressChar/type::0/model::sequence/schemaElement::Employee/type::0/mo
```

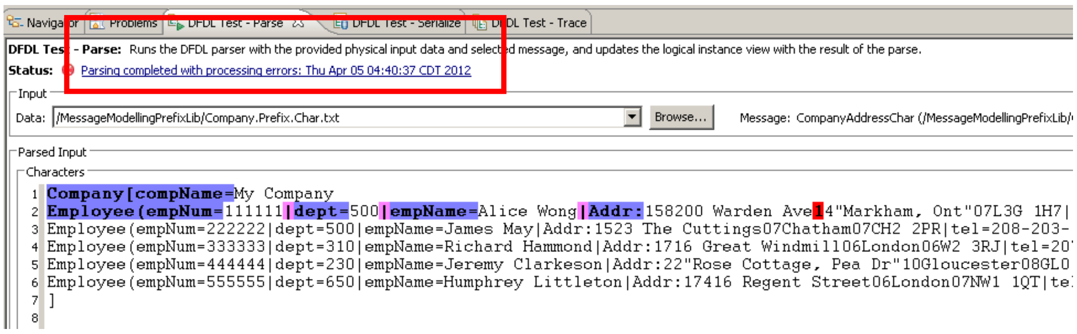
- Errors received during parsing are highlighted in the parsed input section of the DFDL Test - Parse view and hover help is provided.
- To view the trace captured while running the DFDL parser, click the Open DFDL Trace View toolbar button, or click here.
- To view the partial logical instance that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

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.



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: Thu Apr 05 04:40:37 CDT 2012

Input

Data: /MessageModellingPrefixLib/Company.Prefix.Char.txt

Message: CompanyAddressChar (/MessageModellingPrefixLib/

Parsed Input

Characters

```
Company[compName=My Company
Employee(empNum=111111|dept=500|empName=Alice Wong|Addr:158200 Warden Ave14"Markham, Ont"07L3G 1H7|
Employee(empNum=222222|dept=500|empName=James May|Addr:1523 The Cuttings07Chatham07CH2 2PR|tel=208-203-
Employee(empNum=333333|dept=310|empName=Richard Hammond|Addr:1716 Great Windmill06London06W2 3RJ|tel=20
Employee(empNum=444444|dept=230|empName=Jeremy Clarkeson|Addr:22"Rose Cottage, Pea Dr"10Gloucester08GL0
Employee(empNum=555555|dept=650|empName=Humphrey Littleton|Addr:17416 Regent Street06London07NW1 1QT|te
]
```

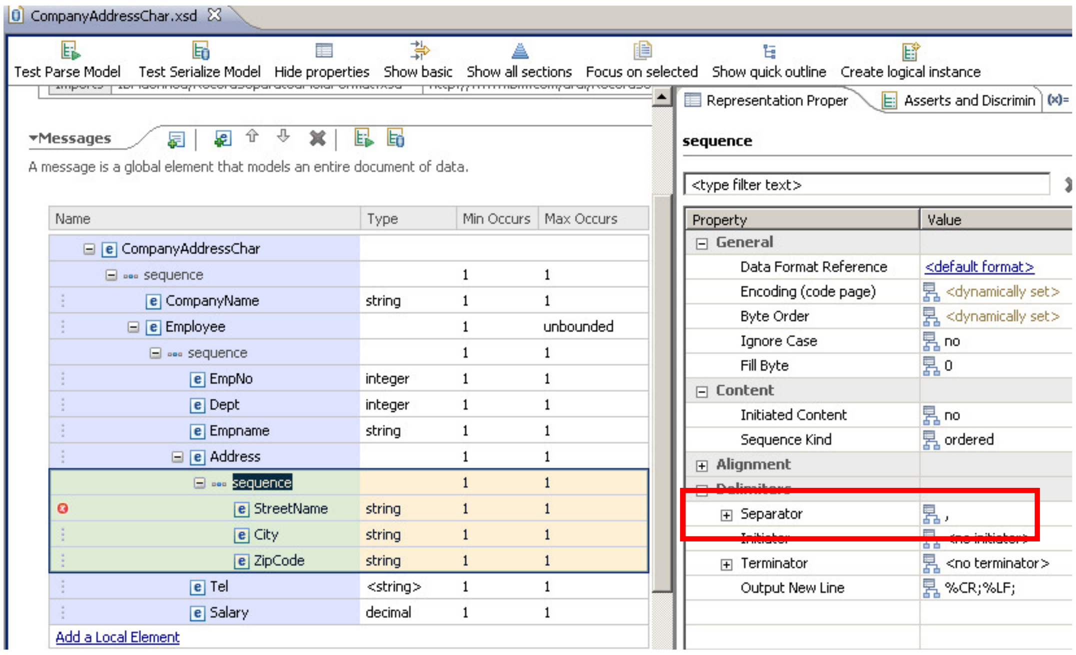
If you can’t work this out, proceed to the next page . . .

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.



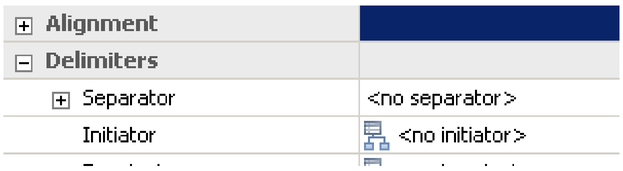
A message is a global element that models an entire document of data.

Name	Type	Min Occurs	Max Occurs
CompanyAddressChar			
sequence		1	1
CompanyName	string	1	1
Employee		1	unbounded
sequence		1	1
EmpNo	integer	1	1
Dept	integer	1	1
Empname	string	1	1
Address		1	1
sequence		1	1
StreetName	string	1	1
City	string	1	1
ZipCode	string	1	1
Tel	<string>	1	1
Salary	decimal	1	1

sequence

Property	Value
General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	no
Fill Byte	0
Content	
Initiated Content	no
Sequence Kind	ordered
Alignment	
Delimiters	
Separator	,
Initiator	<no initiator>
Terminator	<no terminator>
Output New Line	%CR;%LF;

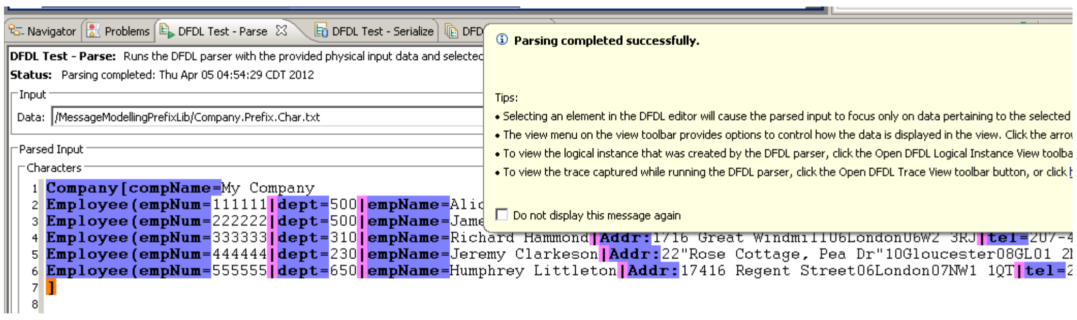
18. Change the separator to "no separator" (use the delete key …. do not set the separator to a blank character).



Alignment	
Delimiters	
Separator	<no separator>
Initiator	<no initiator>

Save the model.

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



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
- The view menu on the view toolbar provides options to control how the data is displayed in the view. Click the arro
- To view the logical instance that was created by the DFDL parser, click the Open DFDL Logical Instance View toolba
- To view the trace captured while running the DFDL parser, click the Open DFDL Trace View toolbar button, or click

Do not display this message again

DFDL Test - Parse: Runs the DFDL parser with the provided physical input data and selected

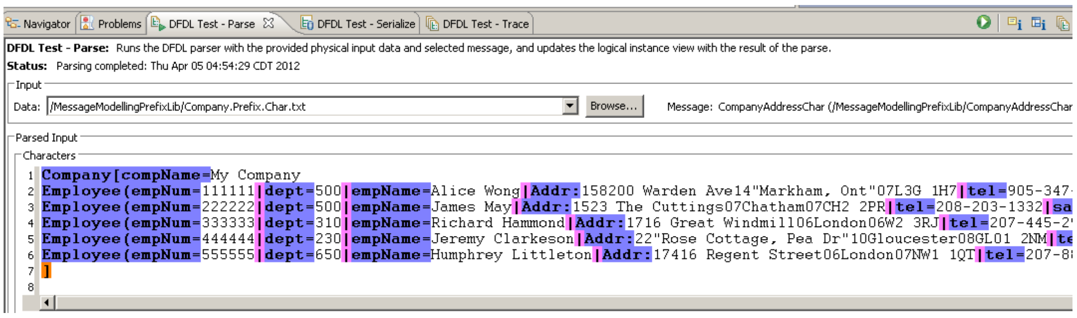
Status: Parsing completed: Thu Apr 05 04:54:29 CDT 2012

Data: /MessageModellingPrefixLib/Company.Prefix.Char.txt

```
Company[compName=My Company
Employee(empNum=111111|dept=500|empName=Alic
Employee(empNum=222222|dept=500|empName=Jame
Employee(empNum=333333|dept=310|empName=Richard Hammond|Addr:1716 Great Windmill06London06W2 3RJ|tel=207-4
Employee(empNum=444444|dept=230|empName=Jeremy Clarkeson|Addr:22"Rose Cottage, Pea Dr"10Gloucester08GL01 2N
Employee(empNum=555555|dept=650|empName=Humphrey Littleton|Addr:17416 Regent Street06London07NW1 1QT|tel=2
]
```

20. Close the yellow completion pop-up.

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



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.

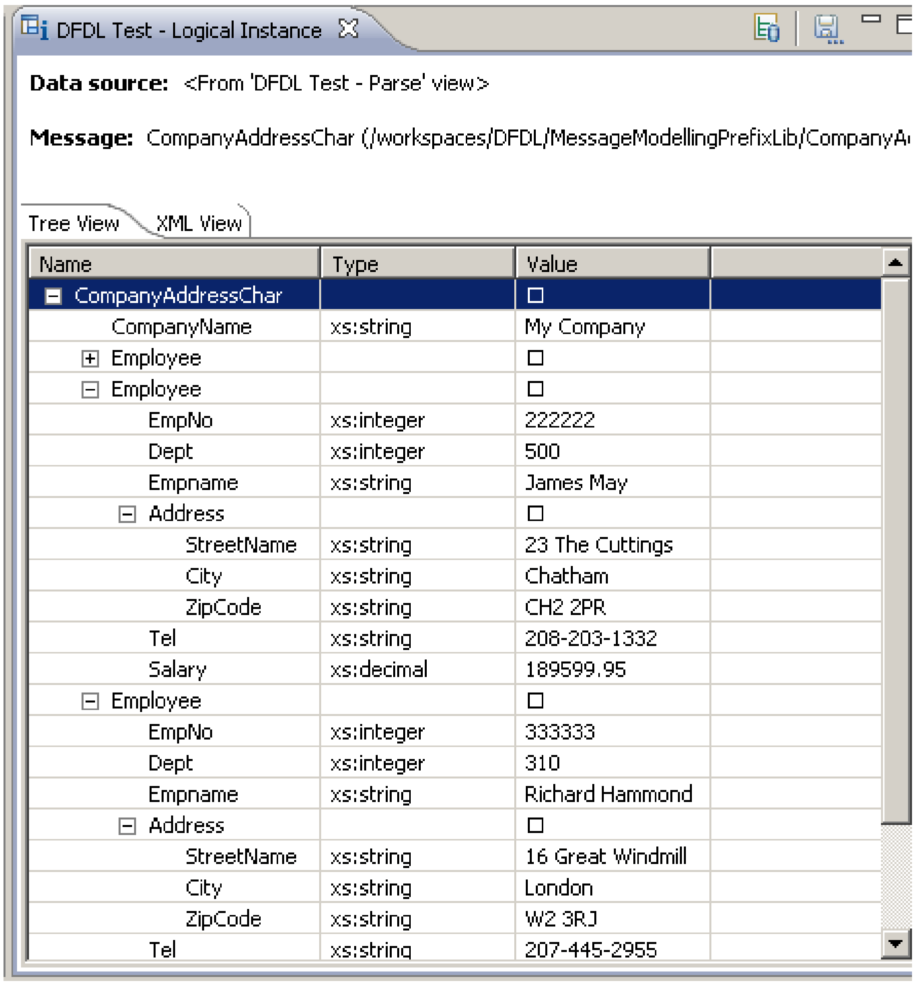
Status: Parsing completed: Thu Apr 05 04:54:29 CDT 2012

Data: /MessageModellingPrefixLib/Company.Prefix.Char.txt

Message: CompanyAddressChar (/MessageModellingPrefixLib/CompanyAddressChar

```
Company[compName=My Company
Employee(empNum=111111|dept=500|empName=Alice Wong|Addr:158200 Warden Ave14"Markham, Ont"07L3G 1H7|tel=905-347-
Employee(empNum=222222|dept=500|empName=James May|Addr:1523 The Cuttings07Chatham07CH2 2PR|tel=208-203-1332|sa
Employee(empNum=333333|dept=310|empName=Richard Hammond|Addr:1716 Great Windmill06London06W2 3RJ|tel=207-445-2
Employee(empNum=444444|dept=230|empName=Jeremy Clarkeson|Addr:22"Rose Cottage, Pea Dr"10Gloucester08GL01 2NM|te
Employee(empNum=555555|dept=650|empName=Humphrey Littleton|Addr:17416 Regent Street06London07NW1 1QT|tel=207-8
]
```

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.



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

Message: CompanyAddressChar (/workspaces/DFDL/MessageModellingPrefixLib/CompanyA

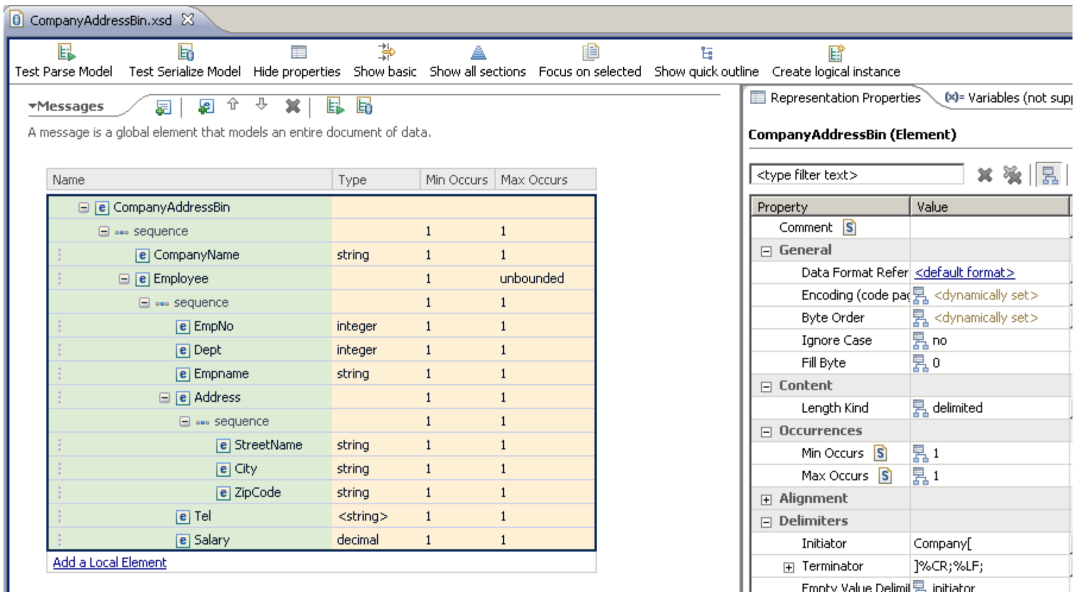
Name	Type	Value
⊟ CompanyAddressChar		☐
CompanyName	xs:string	My Company
⊞ Employee		☐
⊟ Employee		☐
EmpNo	xs:integer	222222
Dept	xs:integer	500
Empname	xs:string	James May
⊟ Address		☐
StreetName	xs:string	23 The Cuttings
City	xs:string	Chatham
ZipCode	xs:string	CH2 2PR
Tel	xs:string	208-203-1332
Salary	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.

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.


CompanyAddressBin.xsd

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

Name	Type	Min Occurs	Max Occurs
CompanyAddressBin			
sequence		1	1
CompanyName	string	1	1
Employee		1	unbounded
sequence		1	1
EmpNo	integer	1	1
Dept	integer	1	1
Empname	string	1	1
Address		1	1
sequence		1	1
StreetName	string	1	1
City	string	1	1
ZipCode	string	1	1
Tel	<string>	1	1
Salary	decimal	1	1

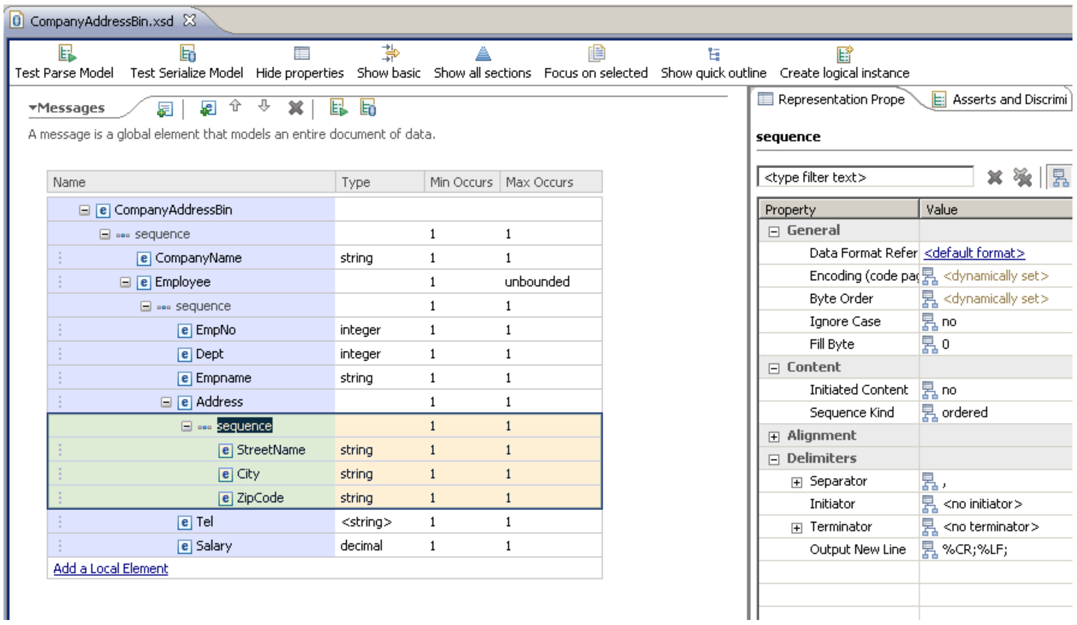
CompanyAddressBin (Element)

<type filter text>

Property	Value
Comment	
General	
Data Format Refer	<default format>
Encoding (code pa	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	no
Fill Byte	0
Content	
Length Kind	delimited
Occurrences	
Min Occurs	1
Max Occurs	1
Alignment	
Delimiters	
Initiator	Company[
Terminator	]%CR;%LF;
Empty Value Delimi	initiator

2. Highlight the Address sequence 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.


CompanyAddressBin.xsd

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

Name	Type	Min Occurs	Max Occurs
CompanyAddressBin			
sequence	string	1	1
CompanyName	string	1	1
Employee		1	unbounded
sequence		1	1
EmpNo	integer	1	1
Dept	integer	1	1
Empname	string	1	1
Address		1	1
sequence		1	1
StreetName	string	1	1
City	string	1	1
ZipCode	string	1	1
Tel	<string>	1	1
Salary	decimal	1	1

sequence

<type filter text>

Property	Value
General	
Data Format Refer	<default format>
Encoding (code pa	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	no
Fill Byte	0
Content	
Initiated Content	no
Sequence Kind	ordered
Alignment	
Delimiters	
Separator	,
Initiator	<no initiator>
Terminator	<no terminator>
Output New Line	%CR;%LF;

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

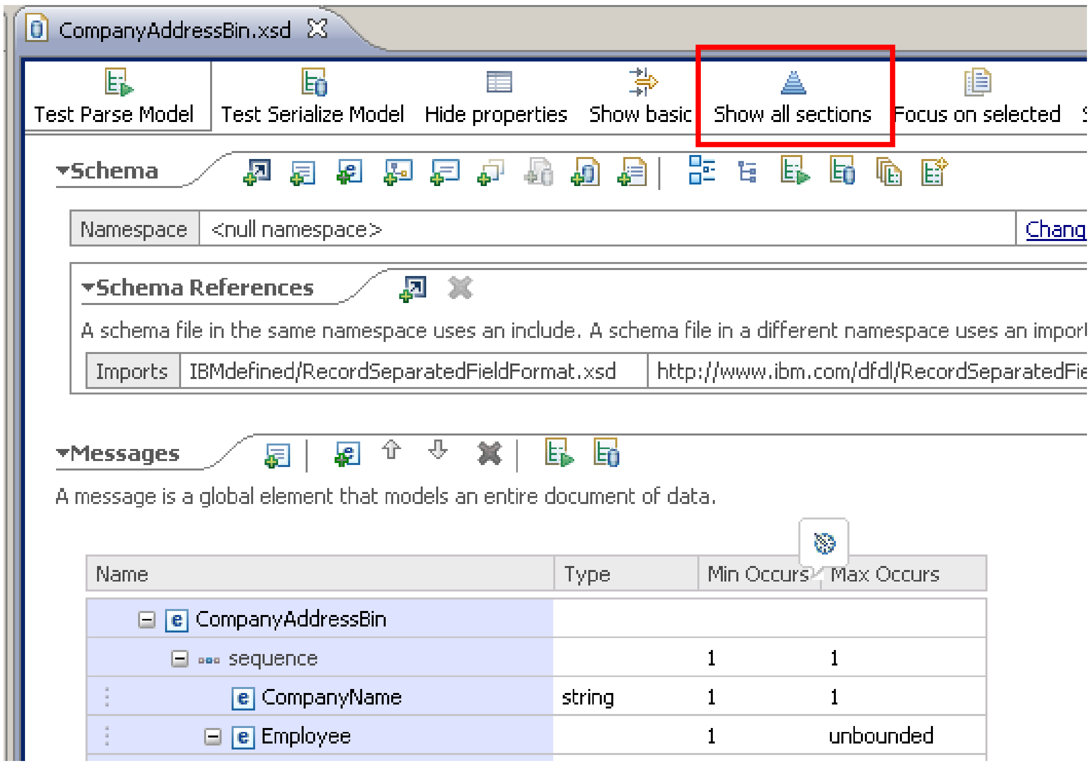
   In this scenario, each of the elements under the Address element will have a prefix of length 2 bytes. The prefix will indicate the length of the element, and the value of the prefix will be a two's complement binary integer. In this case, the value contained in the length prefix will include the length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

   The Address global element may look like this:

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

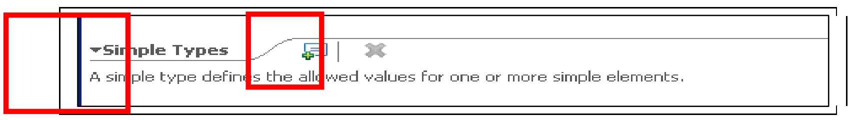
4. To define this type of model, you first need to define a Simple Type. This is used to define the physical characteristics of the prefix length. An element which has a prefix length then simply refers to the simple type.

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


CompanyAddressBin.xsd
Test Parse Model
Test Serialize Model
Hide properties
Show basic
Show all sections
Focus on selected
Schema
Namespace
<null namespace>
Chang
Schema References
A schema file in the same namespace uses an include. A schema file in a different namespace uses an import
Imports
IBMdefined/RecordSeparatedFieldFormat.xsd
http://www.ibm.com/dfdl/RecordSeparatedFie
Messages
A message is a global element that models an entire document of data.

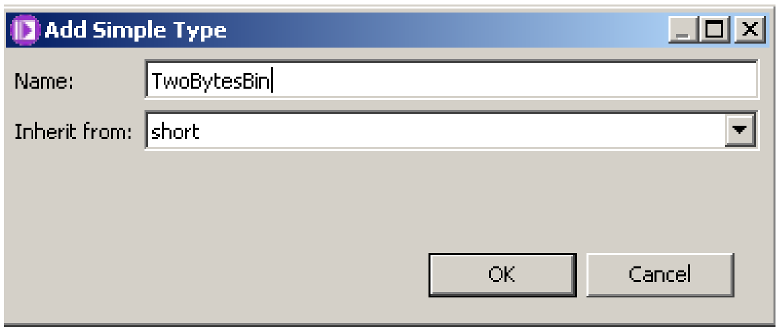
Name	Type	Min Occurs	Max Occurs
CompanyAddressBin			
sequence		1	1
CompanyName	string	1	1
Employee		1	unbounded

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


Simple Types
A simple type defines the allowed values for one or more simple elements.

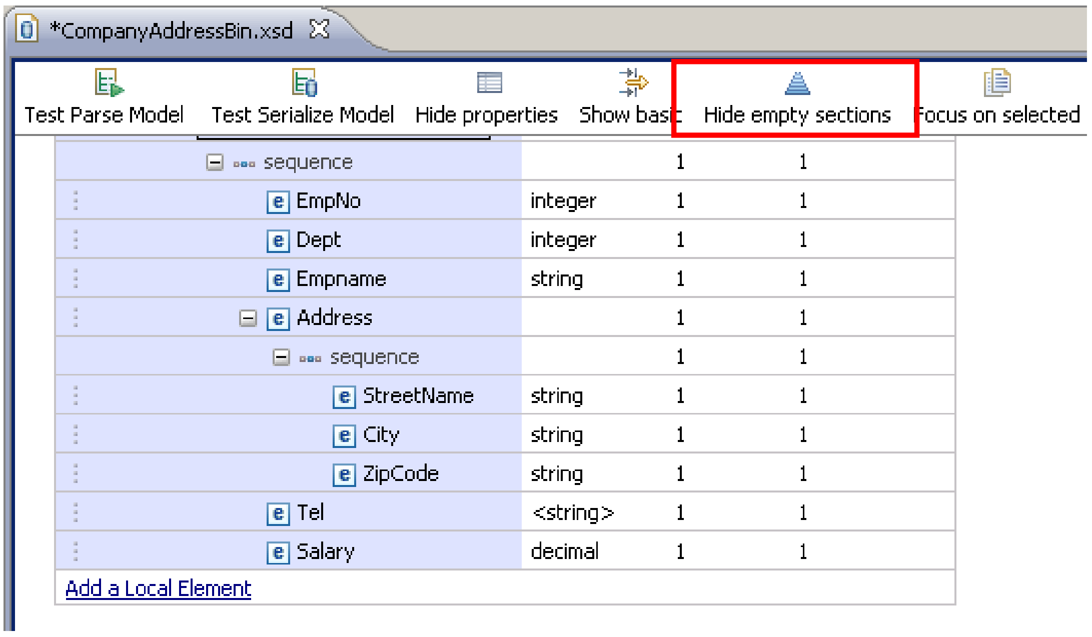
6. In the dialogue window, set Name = TwoBytesBin (you can define your own descriptive name for this type), and set “Inherit from” to short.

   Click OK.



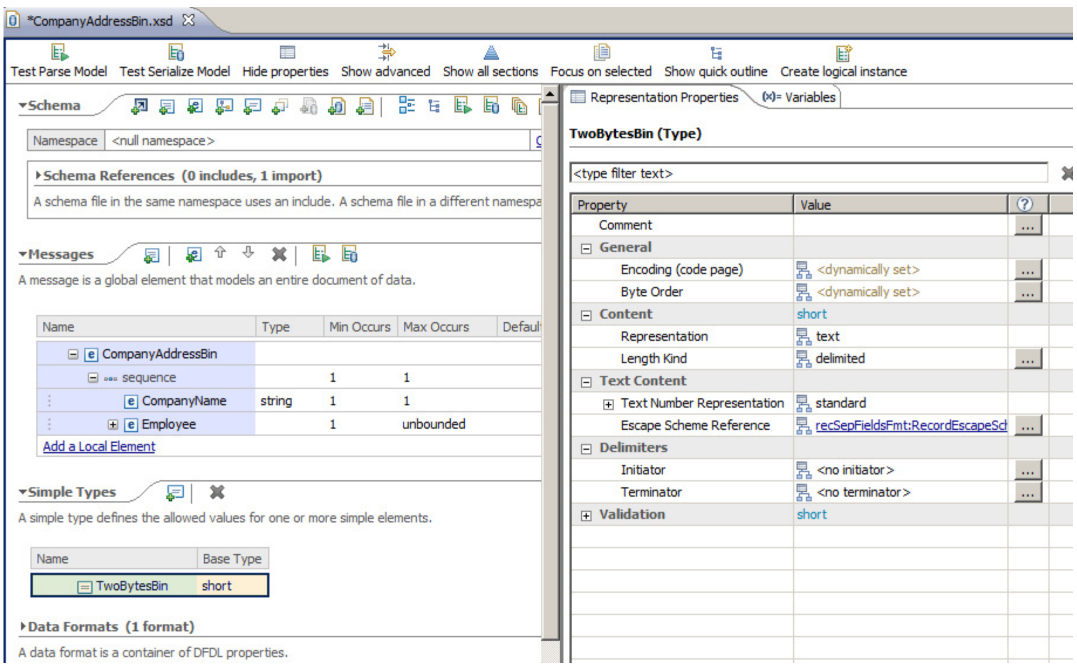
Name: TwoBytesBin
Inherit from: short

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



sequence		1	1
EmpNo	integer	1	1
Dept	integer	1	1
Empname	string	1	1
Address		1	1
sequence		1	1
StreetName	string	1	1
City	string	1	1
ZipCode	string	1	1
Tel	<string>	1	1
Salary	decimal	1	1

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.



A schema file in the same namespace uses an include. A schema file in a different namespa

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

Name	Type	Min Occurs	Max Occurs	Defaul
CompanyAddressBin				
sequence		1	1	
CompanyName	string	1	1	
Employee		1	unbounded	

A simple type defines the allowed values for one or more simple elements.

Name	Base Type
TwoBytesBin	short

A data format is a container of DFDL properties.

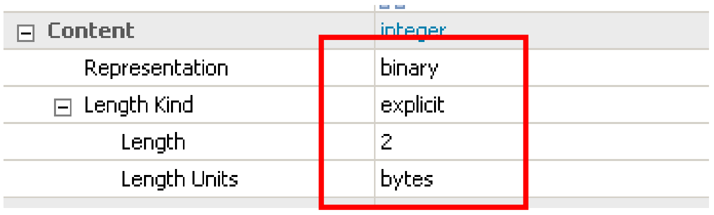
TwoBytesBin (Type)

Property	Value
Comment	
General	
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Content	short
Representation	text
Length Kind	delimited
Text Content	
Text Number Representation	standard
Escape Scheme Reference	recSepFieldsFmt:RecordEscapeSch
Delimiters	
Initiator	<no initiator>
Terminator	<no terminator>
Validation	short

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

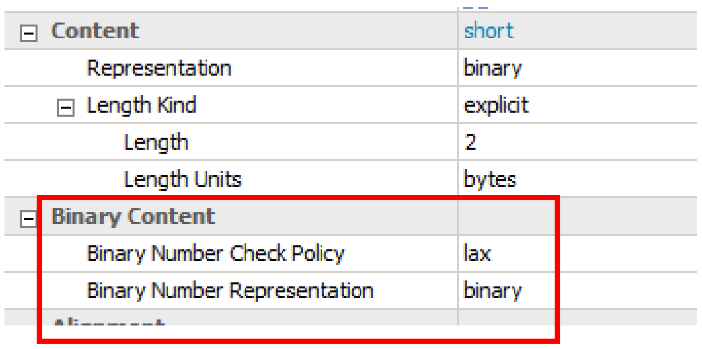
Second, the “Length kind” has been set to “delimited”. Change this to “explicit”.

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



Property	Value
Content	integer
Representation	binary
Length Kind	explicit
Length	2
Length Units	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. And set the Binary Number Check Policy to “lax”. (“Strict” will also work in this example). Remember – you need to “Show Advanced” for this property to be shown.



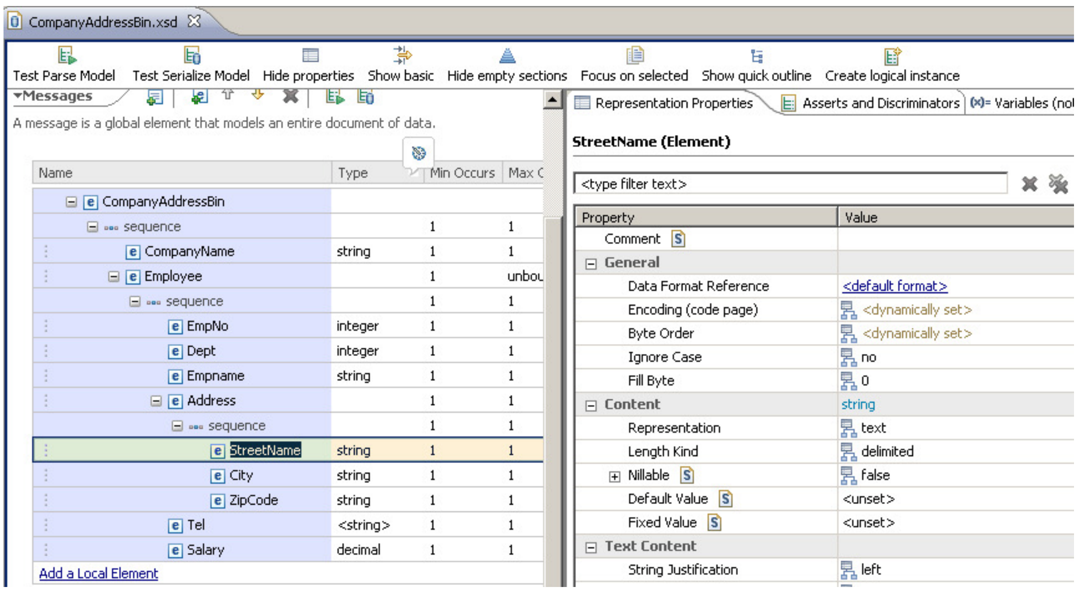
Content	short
Representation	binary
Length Kind	explicit
Length	2
Length Units	bytes
Binary Content	
Binary Number Check Policy	lax
Binary Number Representation	binary

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.



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

Name	Type	Min Occurs	Max C
CompanyAddressBin			
sequence		1	1
CompanyName	string	1	1
Employee		1	unbou
sequence		1	1
EmpNo	integer	1	1
Dept	integer	1	1
Empname	string	1	1
Address		1	1
sequence		1	1
StreetName	string	1	1
City	string	1	1
ZipCode	string	1	1
Tel	<string>	1	1
Salary	decimal	1	1

Add a Local Element

StreetName (Element)

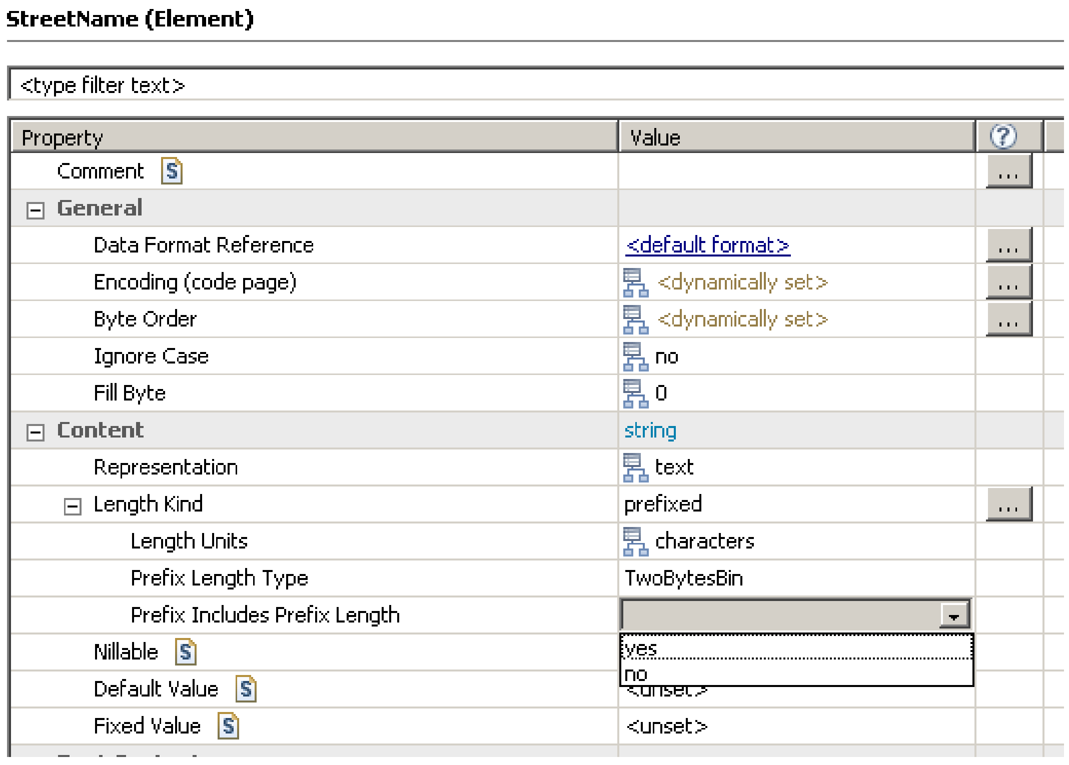
<type filter text>

Property	Value
Comment	
General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	no
Fill Byte	0
Content	string
Representation	text
Length Kind	delimited
Nillable	false
Default Value	<unset>
Fixed Value	<unset>
Text Content	
String Justification	left

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)



StreetName (Element)

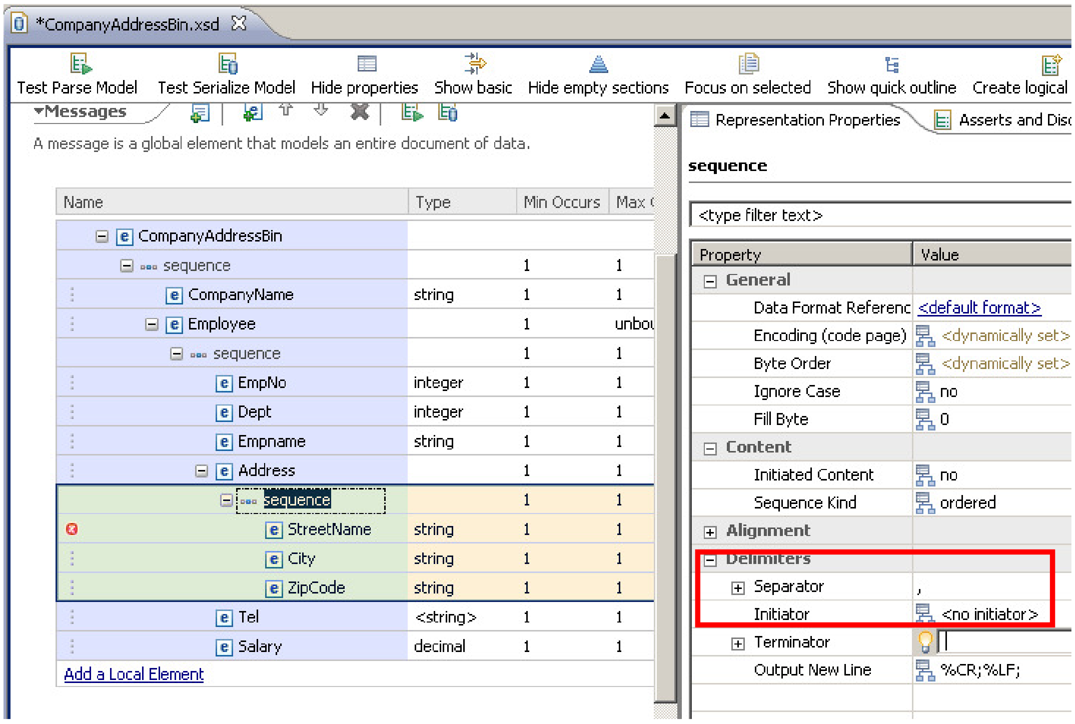
<type filter text>

Property	Value
Comment	
General	
Data Format Reference	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	no
Fill Byte	0
Content	string
Representation	text
Length Kind	prefixed
Length Units	characters
Prefix Length Type	TwoBytesBin
Prefix Includes Prefix Length	yes / no
Nillable	
Default Value	<unset>
Fixed Value	<unset>

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).



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

Name	Type	Min Occurs	Max O
CompanyAddressBin			
sequence		1	1
CompanyName	string	1	1
Employee		1	unbou
sequence		1	1
EmpNo	integer	1	1
Dept	integer	1	1
Empname	string	1	1
Address		1	1
sequence		1	1
StreetName	string	1	1
City	string	1	1
ZipCode	string	1	1
Tel	<string>	1	1
Salary	decimal	1	1

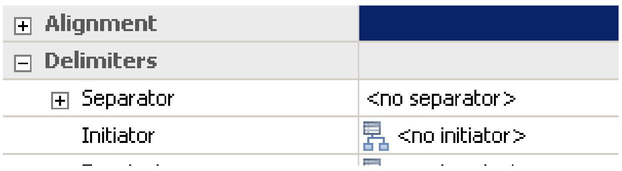
Add a Local Element

sequence

<type filter text>

Property	Value
General	
Data Format Referenc	<default format>
Encoding (code page)	<dynamically set>
Byte Order	<dynamically set>
Ignore Case	no
Fill Byte	0
Content	
Initiated Content	no
Sequence Kind	ordered
Alignment	
Delimiters	
Separator	,
Initiator	<no initiator>
Terminator	
Output New Line	%CR;%LF;

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



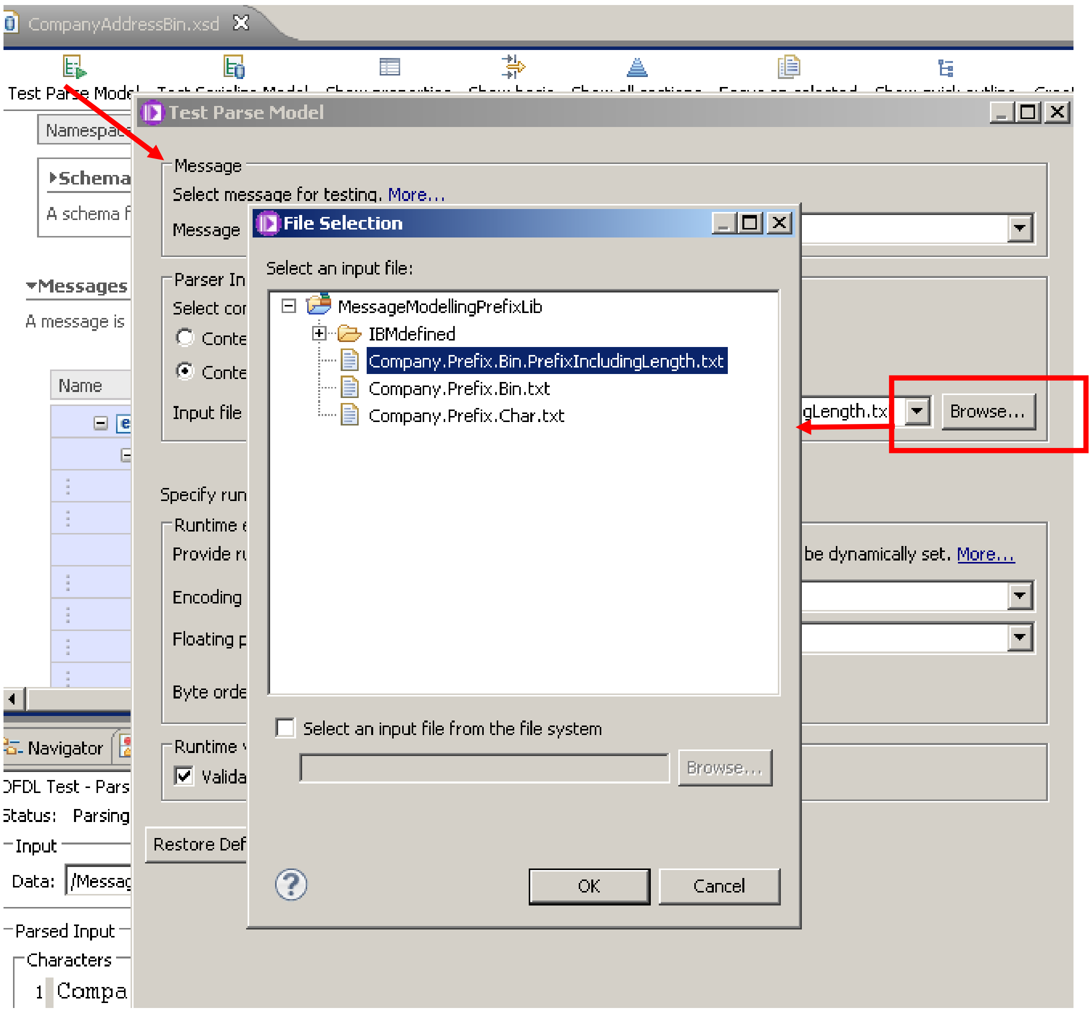
Alignment	
Delimiters	
Separator	<no separator>
Initiator	<no initiator>

Save the model.

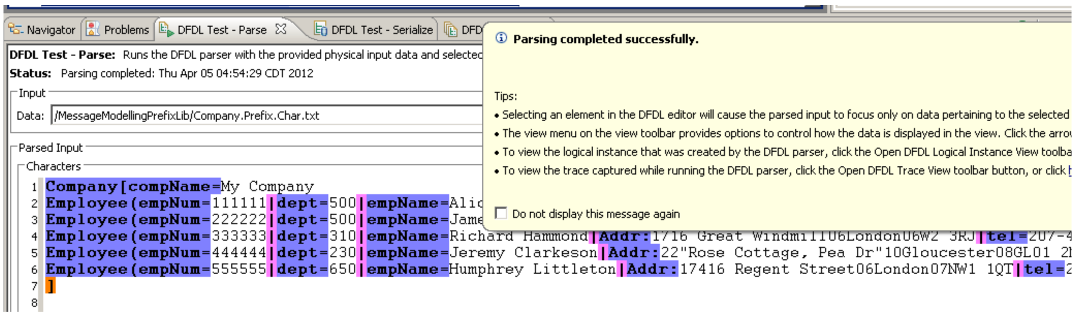
16. Click the Test Parse Model button.

    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.



17. Close the yellow completion pop-up.



DFDL Test - Parse: Runs the DFDL parser with the provided physical input data and selectec

Status: Parsing completed: Thu Apr 05 04:54:29 CDT 2012

Input

Data: /MessageModellingPrefixLib/Company.Prefix.Char.txt

Parsed Input

Characters

```
Company[compName=My Company
Employee(empNum=111111|dept=500|empName=Alic
Employee(empNum=222222|dept=500|empName=Jame
Employee(empNum=333333|dept=310|empName=Richard Hammond|Addr:1716 Great Windmill06London06W2 3RJ|tel=207-4
Employee(empNum=444444|dept=230|empName=Jeremy Clarkeson|Addr:22"Rose Cottage, Pea Dr"10Gloucester08GL01 2
Employee(empNum=555555|dept=650|empName=Humphrey Littleton|Addr:17416 Regent Street06London07NW1 1QT|tel=
]
```

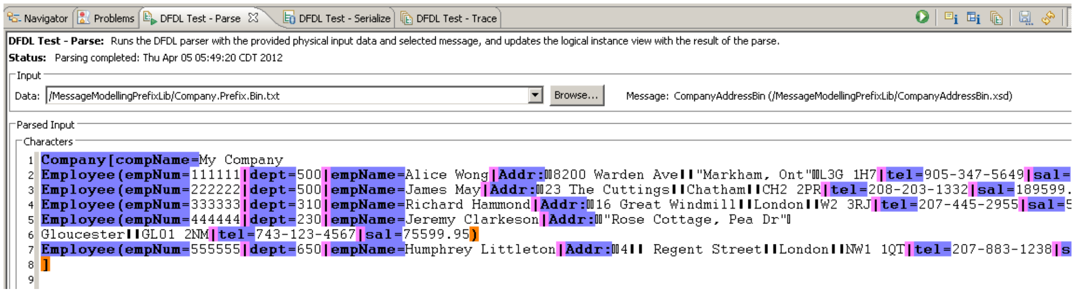
Parsing completed successfully.

Tips:

- Selecting an element in the DFDL editor will cause the parsed input to focus only on data pertaining to the selected
- The view menu on the view toolbar provides options to control how the data is displayed in the view. Click the arro
- To view the logical instance that was created by the DFDL parser, click the Open DFDL Logical Instance View toolba
- To view the trace captured while running the DFDL parser, click the Open DFDL Trace View toolbar button, or click

☐ Do not display this message again

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



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: Thu Apr 05 05:49:20 CDT 2012

Input

Data: /MessageModellingPrefixLib/Company.Prefix.Bin.txt

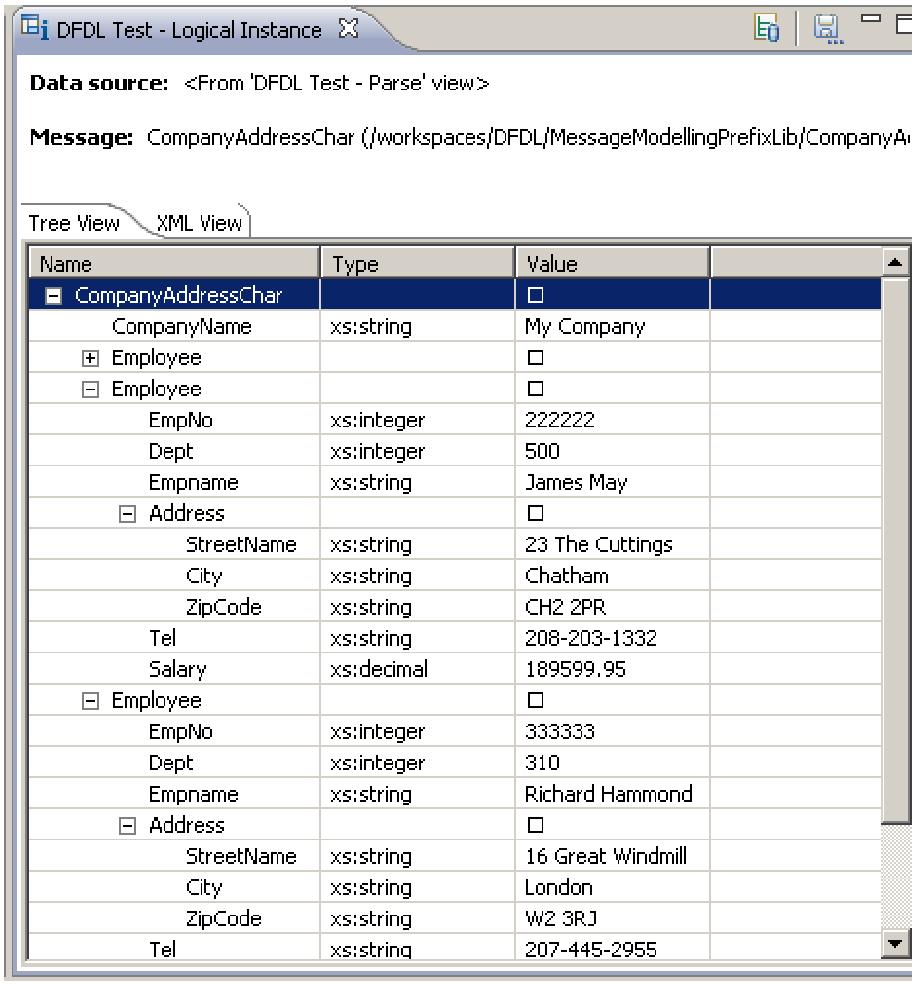
Message: CompanyAddressBin (/MessageModellingPrefixLib/CompanyAddressBin.xsd)

Parsed Input

Characters

```
Company[compName=My Company
Employee(empNum=111111|dept=500|empName=Alice Wong|Addr:▯8200 Warden Ave▮▮"Markham, Ont"▯L3G 1H7|tel=905-347-5649|sal=
Employee(empNum=222222|dept=500|empName=James May|Addr:▯23 The Cuttings▮▮Chatham▮▮CH2 2PR|tel=208-203-1332|sal=189599.
Employee(empNum=333333|dept=310|empName=Richard Hammond|Addr:▯16 Great Windmill▮▮London▮▮W2 3RJ|tel=207-445-2955|sal=5
Employee(empNum=444444|dept=230|empName=Jeremy Clarkeson|Addr:▯"Rose Cottage, Pea Dr"▯
Gloucester▮▮GL01 2NM|tel=743-123-4567|sal=75599.95)
Employee(empNum=555555|dept=650|empName=Humphrey Littleton|Addr:▯4▮▮ Regent Street▮▮London▮▮NW1 1QT|tel=207-883-1238|s
]
```

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.



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

Message: CompanyAddressChar (/workspaces/DFDL/MessageModellingPrefixLib/CompanyA

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 Binary scenario.