



## IBM Integration Bus

### Message Modeling with DFDL

#### Lab 3

#### Record-oriented, tagged, delimited text

**June 2015**

Hands-on lab built at product  
Version 10.0.0.0

<b>1. INTRODUCTION</b> .....	<b>3</b>
1.1 LAB PREPARATION.....	3
1.2 LAB SCENARIO.....	3
<b>2. BUILD THE MESSAGE MODEL</b> .....	<b>4</b>
<b>3. REFINING THE MESSAGE MODEL</b> .....	<b>11</b>
<b>4. TESTING THE MESSAGE MODEL</b> .....	<b>42</b>
<b>END OF LAB GUIDE</b> .....	<b>48</b>

# 1. Introduction

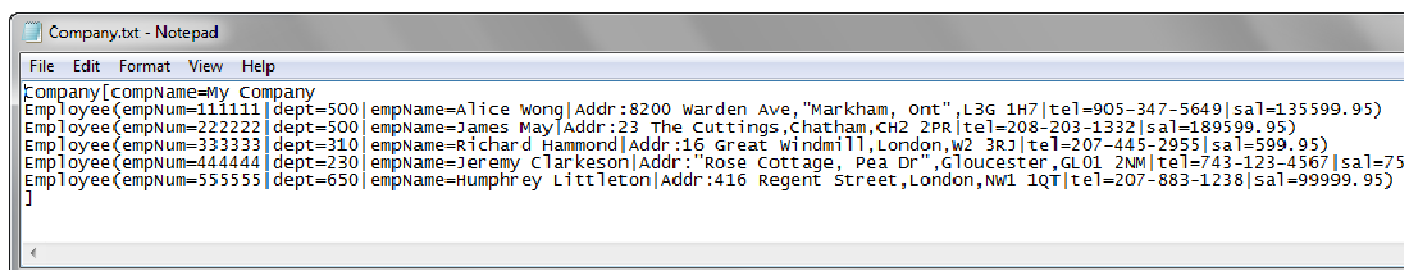
## 1.1 Lab preparation

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

A Record oriented message model is useful to model messages that consist of text strings, but it can also handle binary data. Examples of this type of messages are those that conform to the ACORD AL3, EDIFACT, HL7, SWIFT, or X12 standards. This format allows a high degree of flexibility when defining message formats, and is not restricted to modeling specific industry standards, so you can use it to model your own messages.

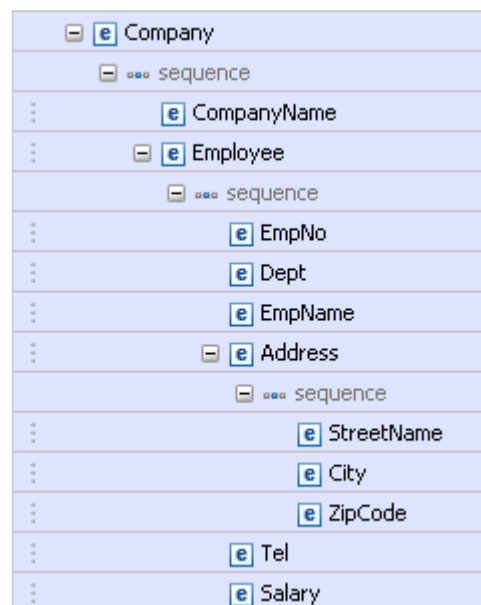
In this lab you will build a message model capable of parsing this tagged / delimited file:



```

Company[compName=My Company
Employee(empNum=11111|dept=500|empName=Alice Wong|Addr:8200 warden Ave,"Markham, Ont",L3G 1H7|tel=905-347-5649|sal=135599.95)
Employee(empNum=22222|dept=500|empName=James May|Addr:23 The Cuttings,Chatham,CH2 2PR|tel=208-203-1332|sal=189599.95)
Employee(empNum=33333|dept=310|empName=Richard Hammond|Addr:16 Great Windmill,London,W2 3RJ|tel=207-445-2955|sal=599.95)
Employee(empNum=44444|dept=230|empName=Jeremy Clarkson|Addr:"Rose Cottage, Pea Dr",Gloucester,GL01 2NM|tel=743-123-4567|sal=750000.95)
Employee(empNum=55555|dept=650|empName=Humphrey Littleton|Addr:416 Regent Street,London,NW1 1QT|tel=207-883-1238|sal=99999.95)
]
  
```

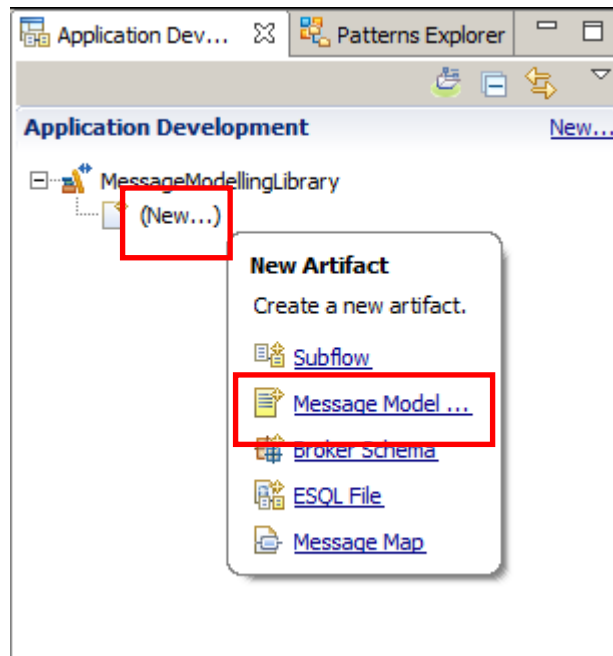
This is an outline of the final message model that you will create:



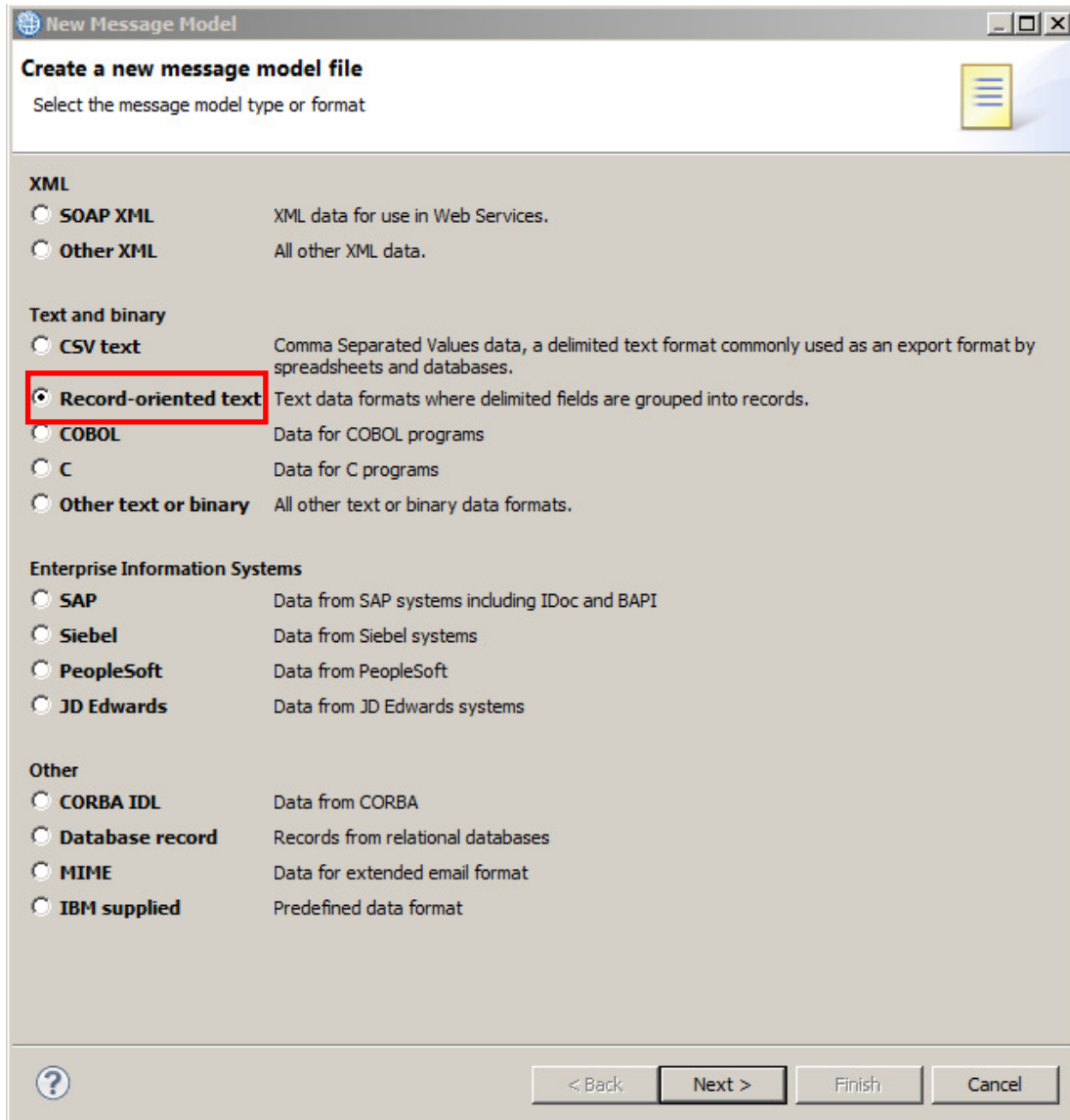
## 2. Build the Message Model

1. In the MessageModellingLibrary that you created in Lab1, click New -> Message Model.

(If you didn't do any earlier labs, you can create a new library called MessageModellingLibrary).



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



**New Message Model**

Create a new message model file

Select the message model type or format

**XML**

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

**Text and binary**

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

**Enterprise Information Systems**

- SAP Data from SAP systems including IDoc and BAPI
- Siebel Data from Siebel systems
- PeopleSoft Data from PeopleSoft
- JD Edwards Data from JD Edwards systems

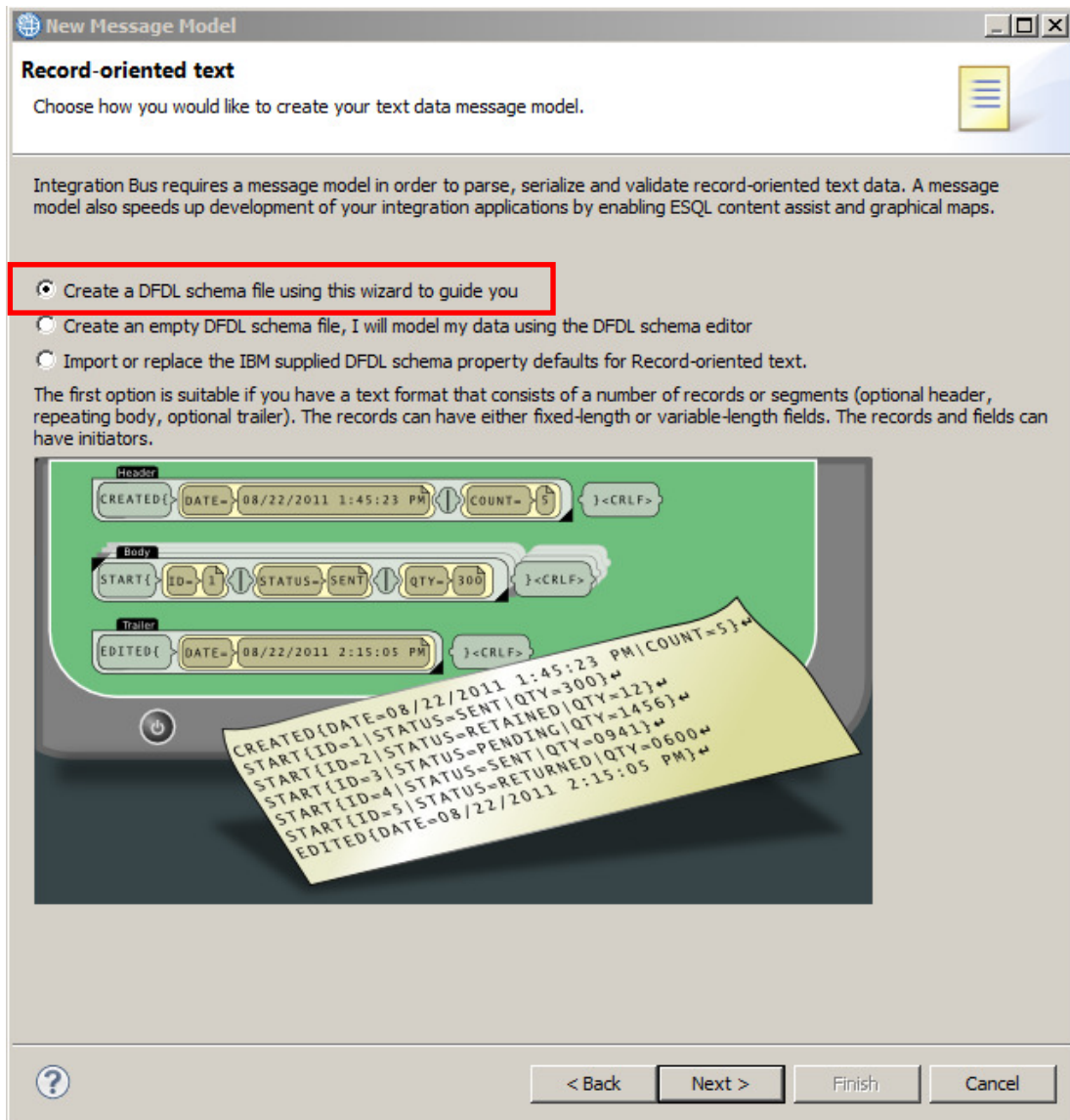
**Other**

- CORBA IDL Data from CORBA
- Database record Records from relational databases
- MIME Data for extended email format
- IBM supplied Predefined data format

? < Back Next > Finish Cancel

3. You can create the new message model using a wizard or create an empty DFDL schema and start from scratch.

Leave the default selection to “use the wizard” and click Next.



4. Enter "Company" as the name for the DFDL Schema and click Next.

**New Message Model**

**Create a Data Format Description Language (DFDL) Schema**  
Specify the location and name of the DFDL schema, and specify the name of the message.

Application or Library:

Folder:

DFDL schema file name:

Message name:

5. Uncheck both "The first record is a header" and "The last record is a trailer".

On the "Body fields" tab, set the Record initiator to "Employee(" and set the number of fields to 6.

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.

The screenshot shows the 'New Message Model' dialog box with the following configuration:

- Record settings:**
  - End of record character: Carriage Return & Line Feed - %CR;%LF;
  - (Blank records will be skipped)
  - The first record is a header
  - The last record is a trailer
- Body fields:**
  - Record initiator: Employee(
  - Number of fields: 6
- Field settings:**
  - Separated by: | - %#124; (UTF-8: 0x7C) (UTF-16: 0x007C)
  - Fixed length:
  - All fields have an initiator:
  - Create default values for fields:
- Encoding code page options:**
  - Dynamic:  (provided to the processor by the application at runtime)
  - Fixed:  US-ASCII
- Global settings:**
  - Escape scheme: Default escape scheme

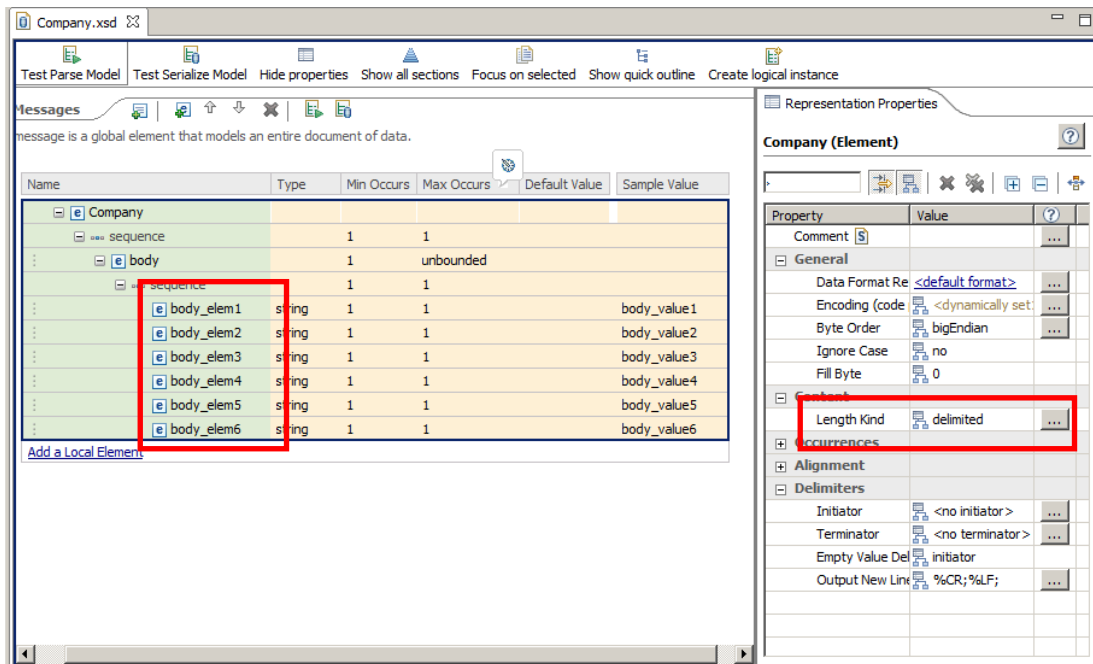
Leave "Separated by" as "|" (pipe) and "All fields have an initiator" checked. These default values match the required for the message model.

Click Finish.

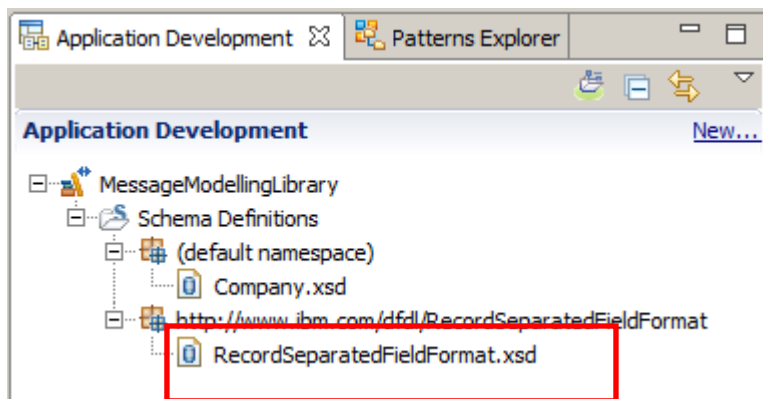


- When the wizard finishes, the DFDL Editor will open with the generated Company.xsd schema file.

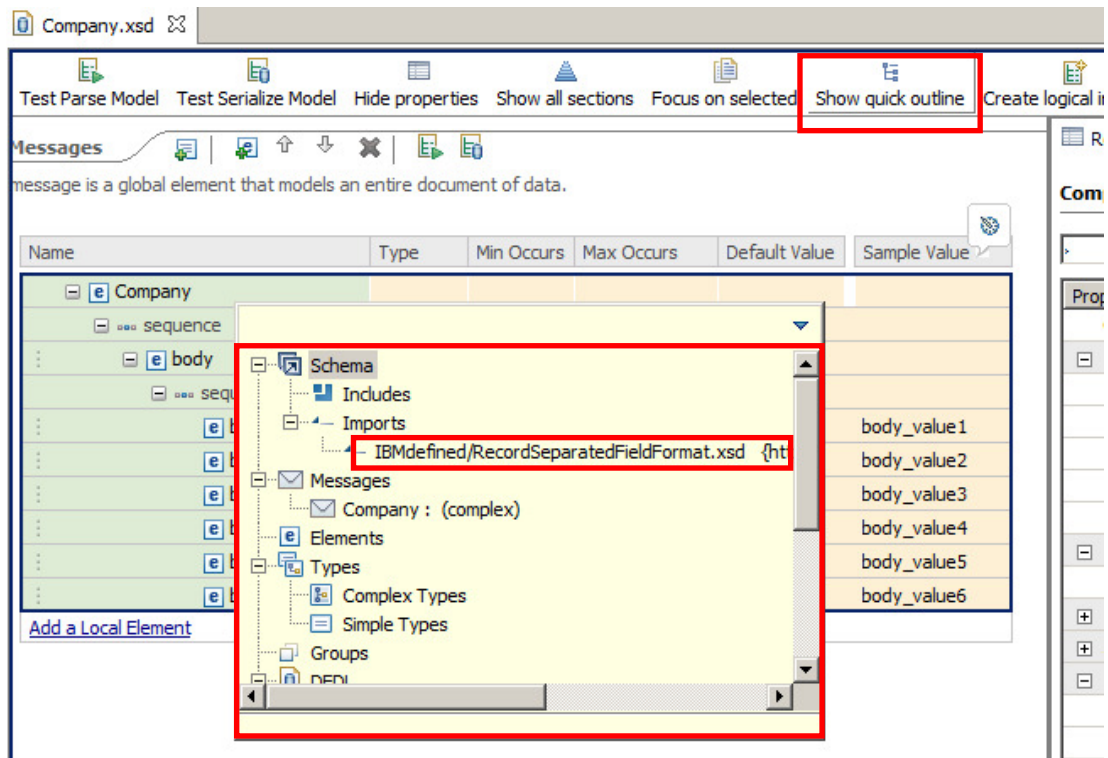
Note that as you defined it in the wizard, the model has six fields, "Length Kind" property is set to delimited.



- The wizard has also generated a second xsd file, RecordSeparatedFieldFormat.xsd. This is the "Helper schema file" that contains the default values for all DFDL properties for the defined Record Oriented data format.



- Click on the "Show quick outline" icon.



The Outline view will appear with a high level view of the elements of your message model. If you click on any of them, the editor will focus on it.

In the outline view you can see that Company.xsd has a reference to the helper schema file RecordSeparatedFieldFormat.xsd.

To close the Outline pop-up, click anywhere else on the editor window.

Hint: if the Messages display "disappears", click Show all Sections (the blue pyramid), and then expand Messages, then expand "body". You can optionally click "Hide Empty Sections" to provide a less cluttered display.



Name	Type	Min Occurs	Max Occurs
[-] e Company			
[-] sequence		1	1
[-] e body		1	unbounded
[-] sequence		1	1
e body_elem1	string	1	1
e body_elem2	string	1	1
e body_elem3	string	1	1
e body_elem4	string	1	1
e body_elem5	string	1	1
e body_elem6	string	1	1

### 3. Refining the Message Model

1. Change the name of the "body" field to "Employee" by single-clicking on it, and overtyping.

Company.xsd

Test Parse Model Test Serialize Model Hide properties Show all sections Focus on selected Show quick outline Create l...

▼Messages

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

Name	Type	Min Occurs	Max Occurs	Default Value	Sample Value
[-] e Company					
[-] ... sequence		1	1		
[-] e Employee		1	unbounded		
[-] ... sequence		1	1		
[-] e body_elem1	string	1	1		body_value1
[-] e body_elem2	string	1	1		body_value2
[-] e body_elem3	string	1	1		body_value3
[-] e body_elem4	string	1	1		body_value4
[-] e body_elem5	string	1	1		body_value5
[-] e body_elem6	string	1	1		body_value6

[Add a Local Element](#)

2. Similarly, change the name of the 6 fields under "Employee" as shown. You can just use the down-arrow to move between element names.

▼Messages


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

Name	Type	Min Occurs	Max Occurs
[-] e Company			
[-] ... sequence		1	1
[-] e Employee		1	unbounded
[-] ... sequence		1	1
[-] e EmpNo	string	1	1
[-] e Dept	string	1	1
[-] e EmpName	string	1	1
[-] e Address	string	1	1
[-] e Tel	string	1	1
[-] e Salary	string	1	1

[Add a Local Element](#)



## 5. Name the new element "CompanyName"


▼Messages 

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

Name	Type	Min Occurs	Max Occurs
[-] [e] Company			
[-] ... sequence		1	1
⋮ [-] [e] Employee		1	unbounded
[-] ... sequence		1	1
⋮ [e] EmpNo	string	1	1
⋮ [e] Dept	string	1	1
⋮ [e] EmpName	string	1	1
⋮ [e] Address	string	1	1
⋮ [e] Tel	string	1	1
⋮ [e] Salary	string	1	1
⋮ [e] CompanyName	string	1	1

[Add a Local Element](#)

## 6. Highlight "CompanyName" and click the yellow "Up" arrow to move this element above the "Employee" element (or you can right-click the element and select "Move Up".)

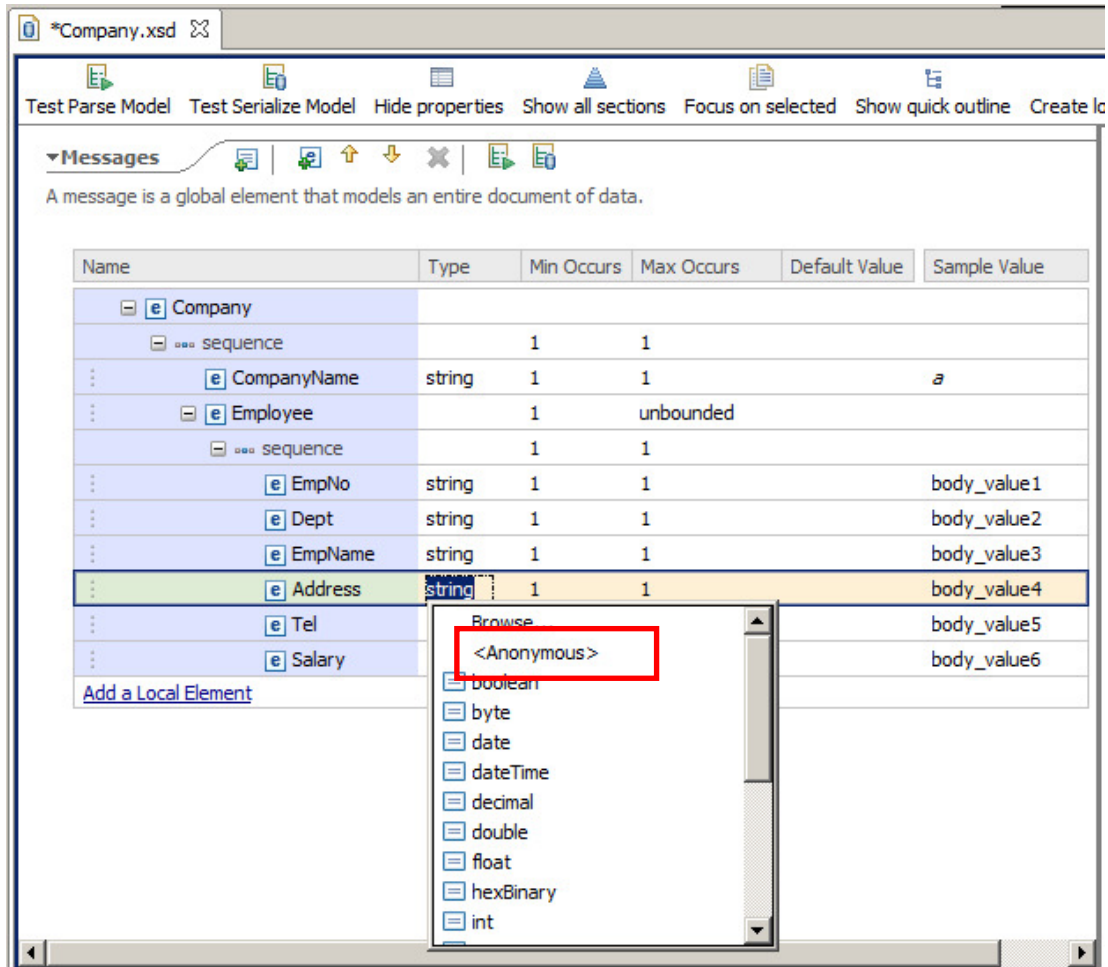
▼Messages 

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

Name	Type	Min Occurs	Max Occurs
[-] [e] Company			
[-] ... sequence		1	1
⋮ [-] [e] Employee		1	unbounded
[-] ... sequence		1	1
⋮ [e] EmpNo	string	1	1
⋮ [e] Dept	string	1	1
⋮ [e] EmpName	string	1	1
⋮ [e] Address	string	1	1
⋮ [e] Tel	string	1	1
⋮ [e] Salary	string	1	1
⋮ [e] CompanyName	string	1	1

[Add a Local Element](#)

- Click on the type column of the Address element and select "<Anonymous>"



The screenshot shows the IBM Integration Bus Message Modelling tool interface. The 'Messages' tab is active, displaying a table of message elements. The 'Address' element is selected, and a 'Browse' dialog is open, showing a list of data types. The '<Anonymous>' type is highlighted with a red box.

Name	Type	Min Occurs	Max Occurs	Default Value	Sample Value
[-] e Company					
[-] sequence		1	1		
...					
e CompanyName	string	1	1		a
...					
[-] e Employee					
[-] sequence		1	1		
...					
e EmpNo	string	1	1		body_value1
...					
e Dept	string	1	1		body_value2
...					
e EmpName	string	1	1		body_value3
...					
e Address	string	1	1		body_value4
...					
e Tel					body_value5
...					
e Salary					body_value6

Available types in the 'Browse' dialog:

- <Anonymous>
- boolean
- byte
- date
- dateTime
- decimal
- double
- float
- hexBinary
- int



- Now add a new element under the new Sequence. Right-click on the Sequence line (although not the text `*** sequence ***` itself), and select Add a Local Element.

Name	Type	Min Occurs	Max Occurs	Default Value
[-] [e] Company				
[-] *** sequence		1	1	
⋮ [e] CompanyName	string	1	1	
⋮ [-] [e] Employee		1	unbounded	
[-] *** sequence		1	1	
⋮ [e] EmpNo	string	1	1	
⋮ [e] Dept	string	1	1	
⋮ [e] EmpName	string	1	1	
⋮ [-] [e] Address		1	1	
[-] *** sequence				
⋮ [e] Tel				
⋮ [e] Salary				
<a href="#">Add a Local Element</a>				

Move Up

Move Down

---

Paste

✖ Delete

---

**➤ Add a Local Element**

➤ Add Complex Local Element...

\*\*\* Add Sequence


⊕ Add Choice

⊕ Add Element Reference...

⊕ Add Group Reference...

Add Hidden Group Reference... (not supported in current version)

- Repeat the previous step twice to add two more fields to the Address element.

▼Messages 


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

Name	Type	Min Occurs	Max Occurs
[-] [e] Company			
[-] *** sequence		1	1
⋮ [e] CompanyName	string	1	1
⋮ [-] [e] Employee		1	unbounded
[-] *** sequence		1	1
⋮ [e] EmpNo	string	1	1
⋮ [e] Dept	string	1	1
⋮ [e] EmpName	string	1	1
⋮ [-] [e] Address		1	1
[-] *** sequence		1	1
⋮ [e] field1	string	1	1
⋮ [e] field2	string	1	1
⋮ [e] field3	string	1	1
⋮ [e] Tel	string	1	1
⋮ [e] Salary	string	1	1
<a href="#">Add a Local Element</a>			



11. Change the names of the 3 elements you've just added to the following by clicking and overwriting with the new names:

1. StreetName
2. City
3. ZipCode

▼Messages 

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

Name	Type	Min Occurs	Max Occurs
[-] [e] Company			
[-] ... sequence		1	1
...			
[e] CompanyName	string	1	1
...			
[-] [e] Employee		1	unbounded
[-] ... sequence		1	1
...			
[e] EmpNo	string	1	1
...			
[e] Dept	string	1	1
...			
[e] EmpName	string	1	1
...			
[-] [e] Address		1	1
[-] ... sequence		1	1
...			
[e] StreetName	string	1	1
...			
[e] City	string	1	1
...			
[e] ZipCode	string	1	1
...			
[e] Tel	string	1	1
...			
[e] Salary	string	1	1
<a href="#">Add a Local Element</a>			


- Click on the <sequence> content of the Address element and take a look at the Delimiters section in the Representation properties.

Notice the inheritance icon next to the Separator field. The Separator for this element was automatically set to "," (comma) because it was inherited from the Helper Schema file (RecordSeparatedFieldFormat.xsd).

The screenshot displays the IBM Integration Bus V10 interface. The main window shows a tree view of the XSD schema for 'Company.xsd'. The 'Address' element is selected, and its 'sequence' child is highlighted with a red box. The right-hand pane shows the 'Representation Properties' for this 'sequence' element. The 'Delimiters' section is expanded, and the 'Separator' field is highlighted with a red box, showing a comma character (','). The 'Separator S' field is set to 'trailingEmpty', and the 'Separator P' field is set to 'infix'. The 'Initiator' field is set to '<no initiator>'. The 'General' section shows 'Data Format Re' as '<default format>', 'Encoding (code)' as '<dynamically set>', 'Byte Order' as 'bigEndian', 'Ignore Case' as 'no', and 'Fill Byte' as '0'. The 'Content' section shows 'Initiated Center' as 'no' and 'Sequence Kind' as 'ordered'. The 'Occurrences' section shows 'Min Occurs' and 'Max Occurs' both set to '1'. The 'Alignment' section is empty. The 'Delimiters' section is expanded, and the 'Separator' field is highlighted with a red box, showing a comma character (',').

Name	Type	Min Occurs	Max Occurs
Company			
sequence		1	1
CompanyName	string	1	1
Employee		1	unbounded
sequence		1	1
EmpNo	string	1	1
Dept	string	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	string	1	1

13. Click on the type column of the "EmpNo" element and select "integer" (not "int").

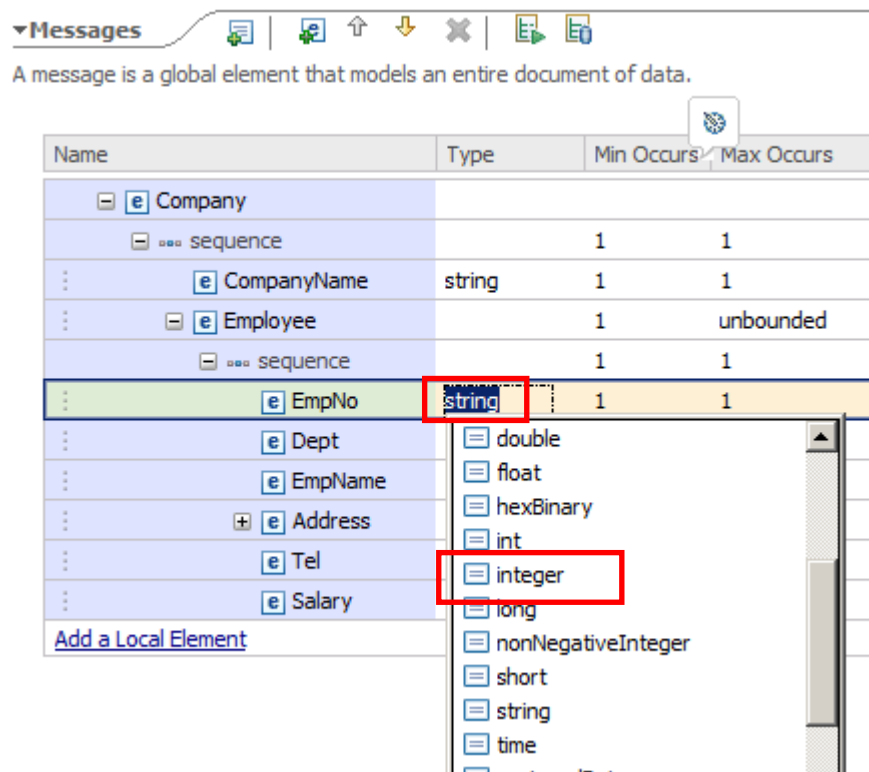
▼ Messages 

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

Name	Type	Min Occurs	Max Occurs
[-] e Company			
[-] ... sequence		1	1
⋮ e CompanyName	string	1	1
⋮ [-] e Employee		1	unbounded
[-] ... sequence		1	1
⋮ e EmpNo	string	1	1
⋮ e Dept			
⋮ e EmpName			
⋮ [+ e Address			
⋮ e Tel			
⋮ e Salary			

[Add a Local Element](#)

- double
- float
- hexBinary
- int
- integer
- long
- nonNegativeInteger
- short
- string
- time



14. Similarly, set the Type of the "Dept" element. = integer (not "int", which would restrict the value to 4 bytes).

15. Set the Type of "Salary" = decimal.

▼Messages

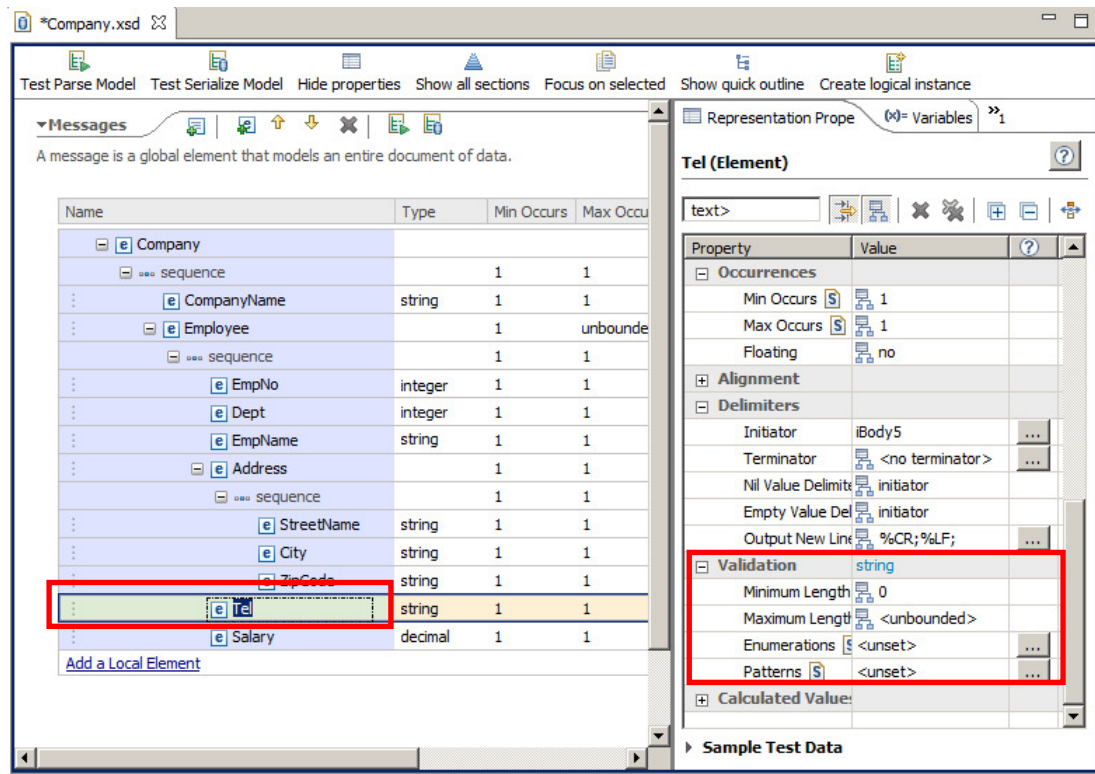
A message is a global element that models an entire document of data.

Name	Type	Min Occurs	Max Occurs
[-] [e] Company			
[-] ... sequence		1	1
...			
[e] CompanyName	string	1	1
...			
[-] [e] Employee		1	unbounded
[-] ... sequence		1	1
...			
[e] EmpNo	integer	1	1
...			
[e] Dept	integer	1	1
...			
[e] EmpName	string	1	1
...			
[-] [e] Address		1	1
[-] ... sequence		1	1
...			
[e] StreetName	string	1	1
...			
[e] City	string	1	1
...			
[e] ZipCode	string	1	1
...			
[e] Tel	string	1	1
...			
[e] Salary	string	1	1

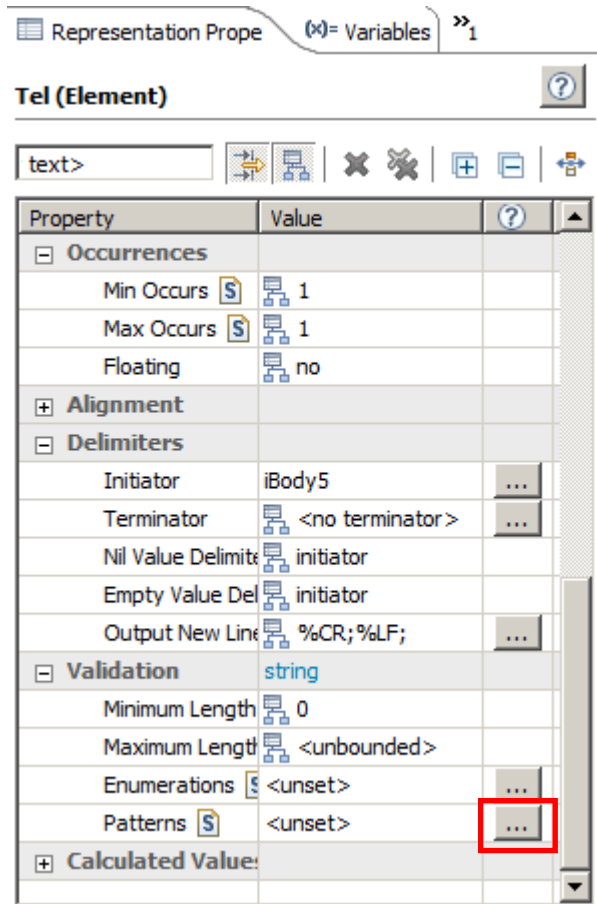
Add a Local Element

- byte
- date
- dateTime
- decimal

- Highlight the "Tel" element and look for the "Validation" section in the Representation Properties view of the DFDL Editor.



17. Click on the "..." button next to the "Patterns" property.



- Click the "Add.." button to create a regular expression that will define a telephone number pattern.

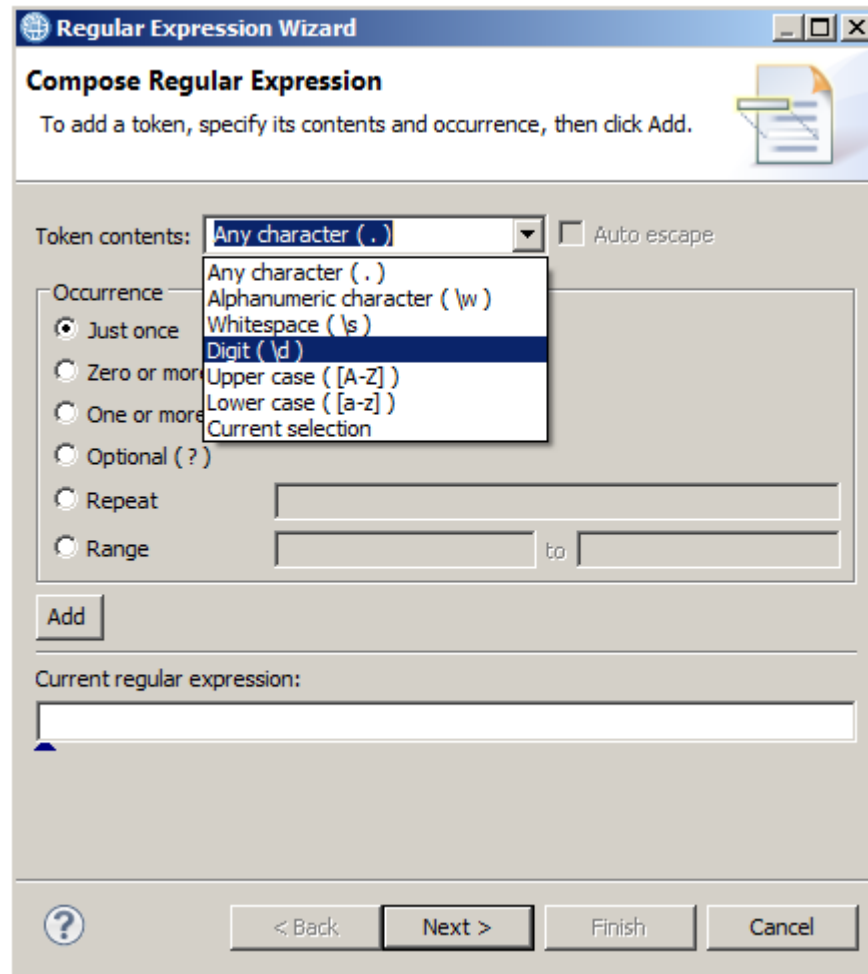
The screenshot shows the 'Property Wizard' dialog box, specifically the 'Pattern' page. The title bar reads 'Property Wizard'. The main heading is 'Pattern'. Below the heading is a descriptive text: 'Set the overall effective pattern by adding parts to the pattern. The pattern can be tested by typing example text. Note that patterns defined here must follow XML Schema regular expression rules.'

The dialog is divided into several sections:

- Effective pattern (more...):** A text input field.
- Test Pattern:** A section containing:
  - Example text:** A text input field.
  - Does example match pattern?:** A text input field.
- Define Pattern:** A section containing:
  - Part of pattern inherited from parent chain:** A text input field.
  - Part of pattern defined on this type:** A text input field.
  - A large empty list box.
  - A vertical stack of buttons on the right side of the list box: 'Add...' (highlighted with a red box), 'Edit...', 'Remove', 'Up', and 'Down'.

At the bottom of the dialog, there is a help icon (a question mark in a circle) on the left, and 'Finish' and 'Cancel' buttons on the right.

19. In the Regular Expression wizard, select "Digit" from the "Token contents" dropdown.





20. Then select the "Repeat" option, enter "3" as its value and click "Add".

**Regular Expression Wizard**

**Compose Regular Expression**

To add a token, specify its contents and occurrence, then click Add.

Token contents:   Auto escape

Occurrence

Just once

Zero or more (\*)

One or more (+)

Optional (?)

Repeat

Range  to

Current regular expression:

21. In the "Current regular expression" field, enter a hyphen ("-") after the text:

**Regular Expression Wizard**

**Compose Regular Expression**

To add a token, specify its contents and occurrence, then click Add.

Token contents:   Auto escape

Occurrence

Just once

Zero or more ( \* )

One or more ( + )

Optional ( ? )

Repeat

Range  to

Current regular expression:

22. Make sure the "token contents" dropdown is set to "Digit" and click the "Add" button again, to add another 3 digits expression.

**Regular Expression Wizard**

**Compose Regular Expression**

To add a token, specify its contents and occurrence, then click Add.

Token contents: **Digit (\d)**  Auto escape

Occurrence:

- Just once
- Zero or more (\*)
- One or more (+)
- Optional (?)
- Repeat: 3
- Range: to

**Add**

Current regular expression: **\d{3}-d{3}**

23. In the "Current regular expression" field, enter another hyphen ("-") after the text:

**Regular Expression Wizard**

**Compose Regular Expression**

To add a token, specify its contents and occurrence, then click Add.

Token contents:   Auto escape

Occurrence

Just once

Zero or more ( \* )

One or more ( + )

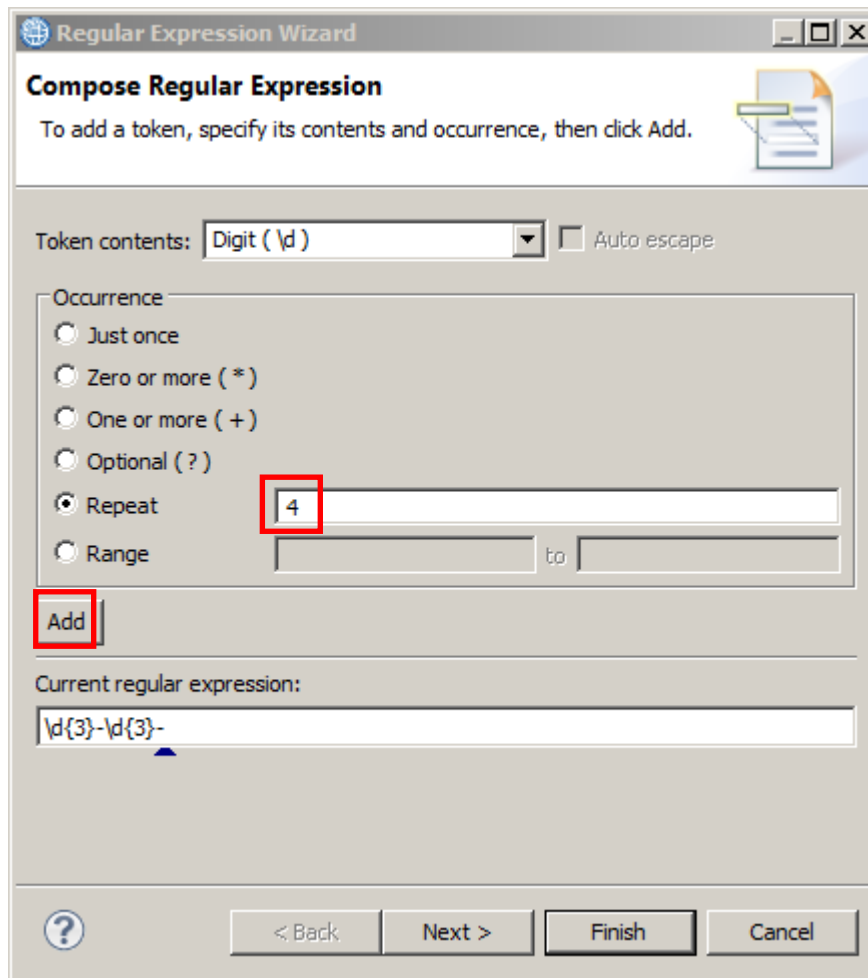
Optional ( ? )

Repeat

Range  to

Current regular expression:

24. Make sure the "token contents" dropdown is set to "Digit", modify the "Repeat" field from "3" to "4" and click the "Add" button again, to add a 4 digits expression.



**Regular Expression Wizard**

**Compose Regular Expression**

To add a token, specify its contents and occurrence, then click Add.

Token contents:   Auto escape

Occurrence

Just once

Zero or more ( \* )

One or more ( + )

Optional ( ? )

Repeat

Range  to

Current regular expression:

25. Click on the Next button.

**Regular Expression Wizard**

**Compose Regular Expression**

To add a token, specify its contents and occurrence, then click Add.

Token contents:   Auto escape

Occurrence

Just once

Zero or more ( \* )

One or more ( + )

Optional ( ? )

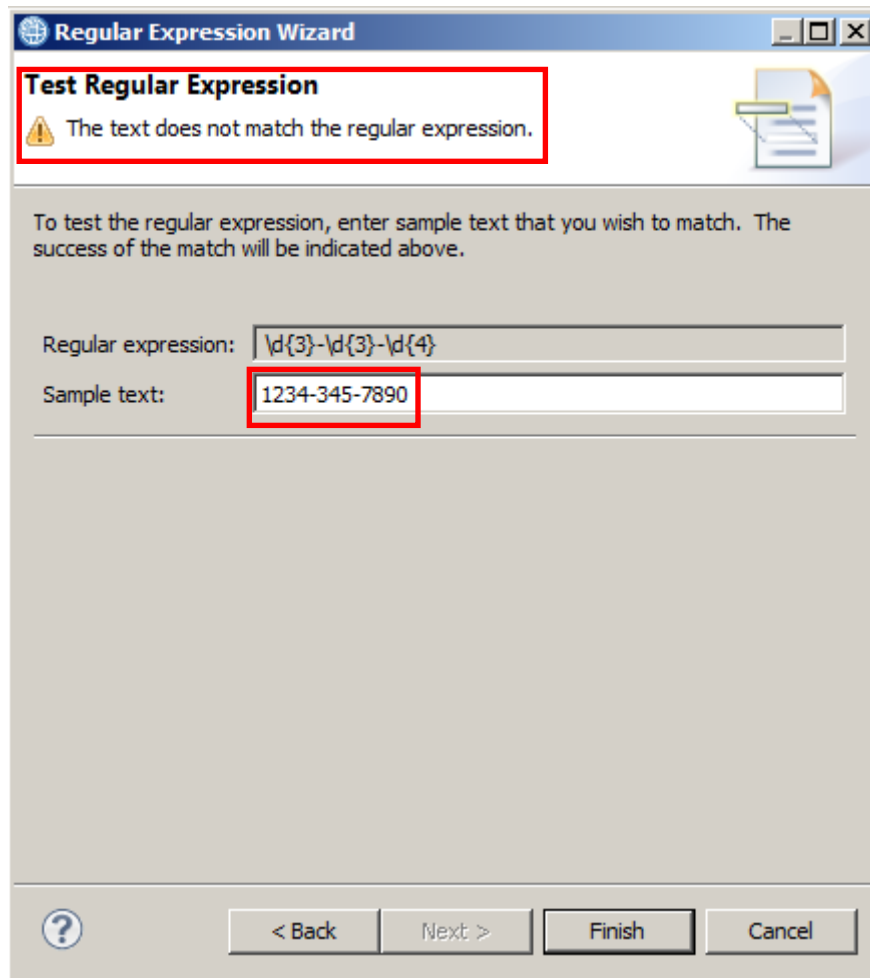
Repeat

Range  to

Current regular expression:

26. The Regular Expression Wizard has a testing feature that lets you validate the regular expression you've just built.

Enter different strings to test the regular expression, and check that the only valid format is: "3 digits - 3 digits - 4 digits".



Then click the Finish button.

27. The Property Wizard is a powerful tool that allows you to build complex regular expressions. In this case you've added just one, but you could create a more complex one by adding several expressions.

Click the Finish button.

**Property Wizard**

**Pattern**

Set the overall effective pattern by adding parts to the pattern. The pattern can be tested by typing example text. Note that patterns defined here must follow XML Schema regular expression rules.

Effective pattern ([more...](#)): `\d{3}-\d{3}-\d{4}`

**Test Pattern**

Example text:

Does example match pattern?

**Define Pattern**

Part of pattern inherited from parent chain:

Part of pattern defined on this type: `\d{3}-\d{3}-\d{4}`


`\d{3}-\d{3}-\d{4}`

Add...  
Edit...  
Remove  
Up  
Down

Finish Cancel



28. Notice that the "Tel" element's type has changed from "string" to "<string>", an anonymous local restriction of xs:string, in order to carry the pattern facet.

▼ Messages 

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

Name	Type	Min Occurs	Max Occurs
[-] e Company			
[-] ... sequence		1	1
⋮ e CompanyName	string	1	1
⋮ [-] e Employee		1	unbounded
[-] ... sequence		1	1
⋮ e EmpNo	integer	1	1
⋮ e Dept	integer	1	1
⋮ e EmpName	string	1	1
⋮ [-] e Address		1	1
[-] ... sequence		1	1
⋮ e StreetName	string	1	1
⋮ e City	string	1	1
⋮ e ZipCode	string	1	1
⋮ e Tel	<string>	1	1
⋮ e Salary	decimal	1	1

[Add a Local Element](#)

29. Change the "Default Value" of the "Tel" element to a pattern complying value by double-clicking on the "Default Value" row in the Representation Properties (for example: 999-999-9999).

Representation Properties (x) Variables Asserts and Discriminators

**Tel (Element)**

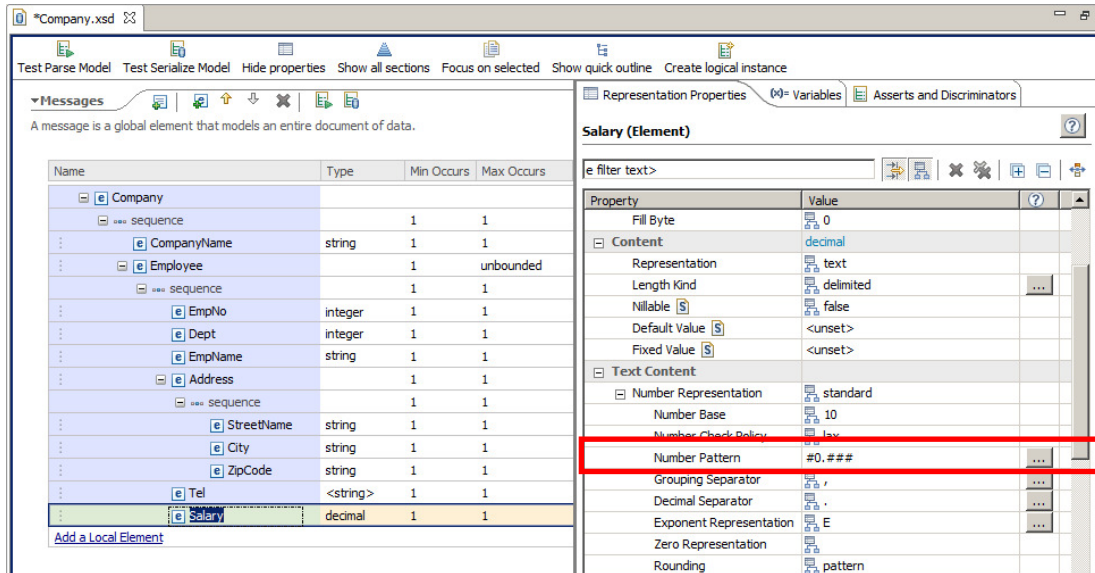
e filter text>

Property	Value	?
Comment <span>S</span>		...
<b>General</b>		
Data Format Reference	<default format>	...
Encoding (code page)	<dynamically set>	...
Byte Order	bigEndian	...
Ignore Case	no	
Fill Byte	0	
<b>Content</b>	string	
Representation	text	
Length Kind	delimited	...
Nilable <span>S</span>	false	
Default Value <span>S</span>	999-999-9999	
Fixed Value <span>S</span>	<unset>	
<b>Text Content</b>		
String Justification	left	
String Pad Character	%SP;	
Truncate Specified Length Strin	no	
Pad Kind	none	
Trim Kind	padChar	

30. Now highlight the "Salary" element and look for the "Text Content" section in the Representation Properties of the DFDL Editor.

Expand "Number Representation".

Click on the button (three dots) next to "Number Pattern".



31. In the Number Pattern Property Wizard, change the Pattern to "#0.##" (delete the final #).

Enter "1234.1234" in the "Number" field in the Text Format section. Click on the "Apply Pattern" button to test the Number Pattern. Notice that the number was changed from "1234.1234" to "1234.12" to comply with the defined number pattern.

**Property Wizard**  
**Number Pattern**  
Set and test values for the number pattern properties.

Pattern describes the format of the text number. Click [here](#) to see symbols and meanings.

**Pattern:** #0.##

**Text Format**  
Number Type: decimal  
Number: 1234.1234 **Apply Pattern**  
Formatted: 1234.12 **Extract Pattern**

**Standard number** A number is represented as characters in the 'encoding' code page. [More...](#)

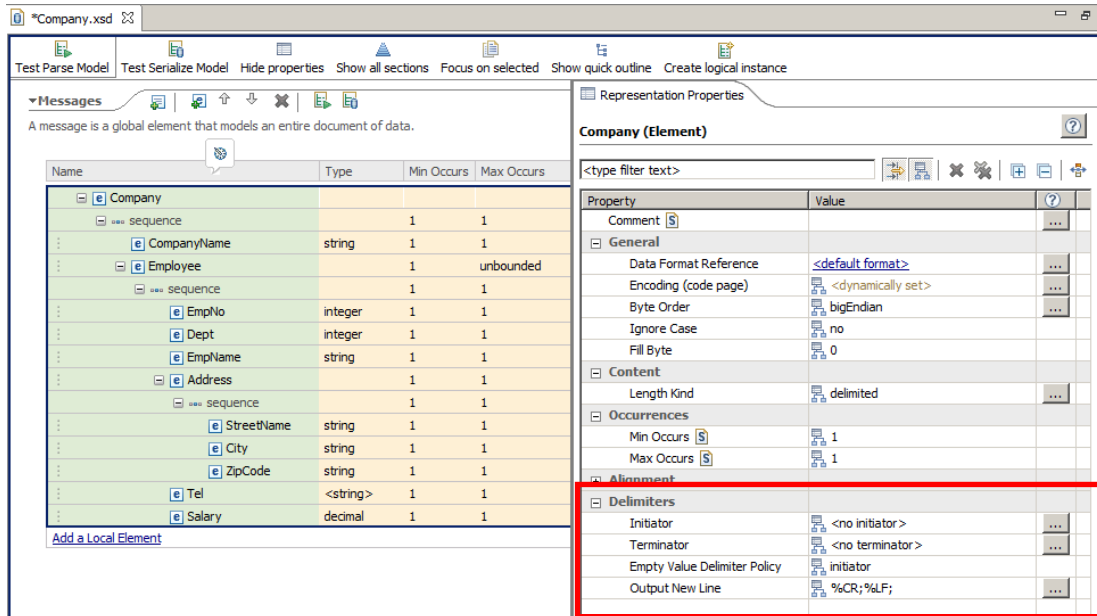
Number base: 10  
Number check policy: lax  
Grouping separator: ,  
Decimal separator: .  
Exponent character: E  
Infinity representation character: Inf  
NaN representation character: NaN  
Zero representation:  
Number rounding: pattern  
Number rounding increment:  
Number rounding mode: roundUp

**Finish** **Cancel**

Click Finish.

32. Next you will define the Initiators, Terminators and Separators for the Message Model.

Click on the "Company" element (message root) and look at the "Delimiters" section in the Representation properties view in the DFDL Editor.



33. Enter "Company[" as the Initiator, and "]%CR;%LF;" as the terminator. (Do not include the quotation marks).

Hint: after you have entered the "]", you can use Ctrl-Space to use the Toolkit Content Assist editor, and select the CR and LF values.

This definition implies that the record starts with a "Company[" tag and ends with a "]%CR;%LF;" tag.

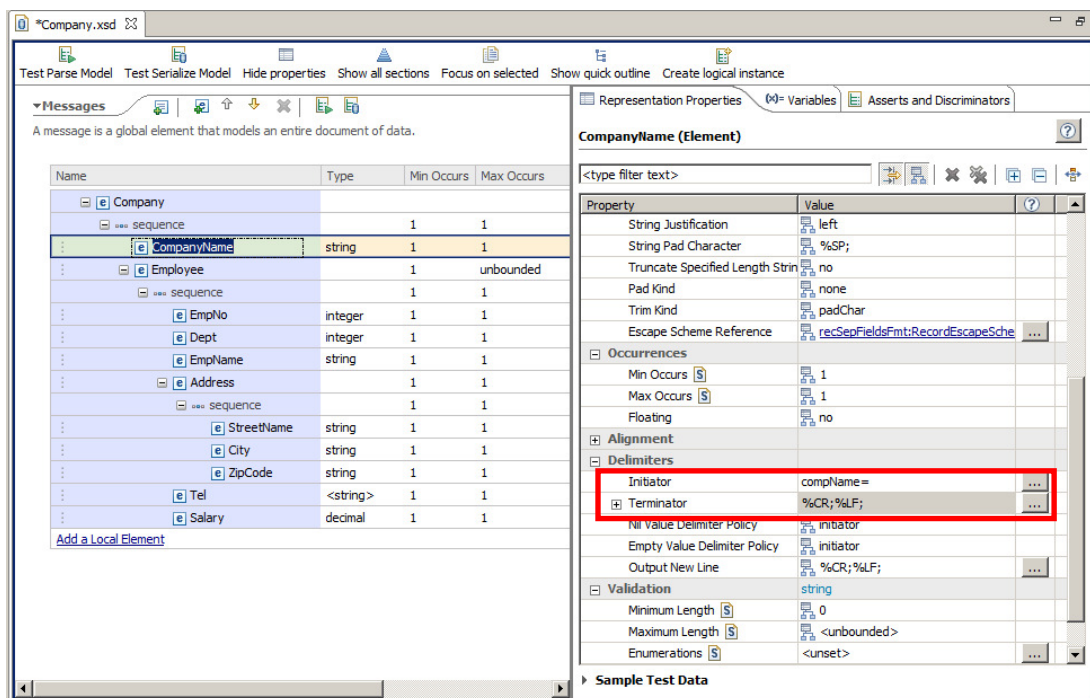
Representation Properties

**Company (Element)**

<type filter text>

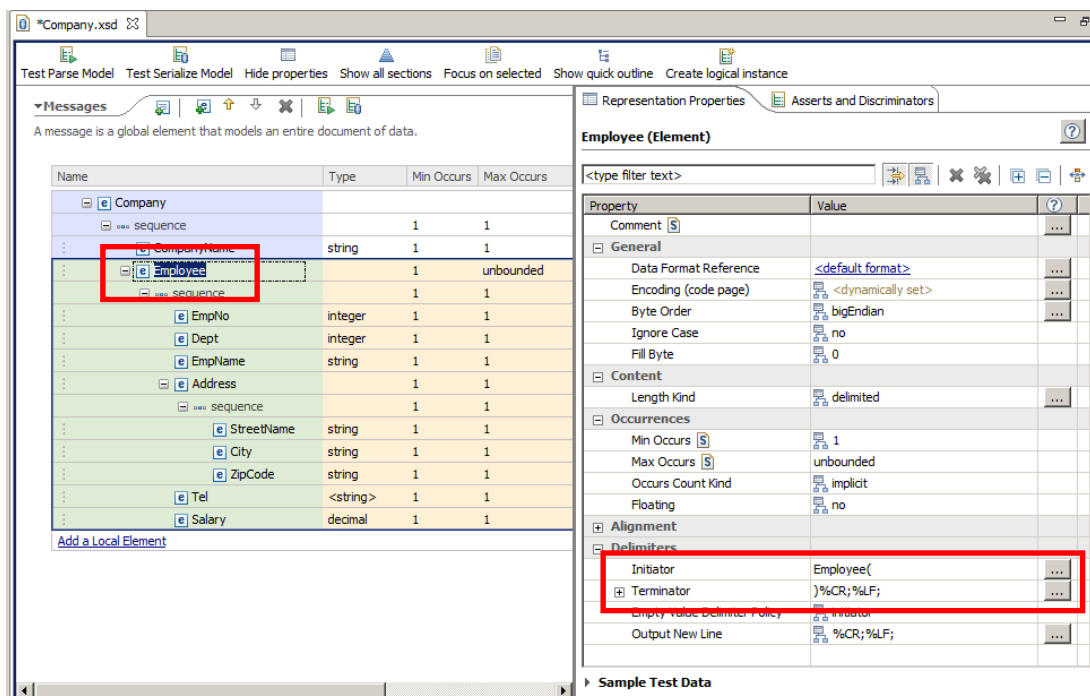
Property	Value	
Comment <input type="text" value="S"/>		...
<b>General</b>		
Data Format Reference	<default format>	...
Encoding (code page)	<dynamically set>	...
Byte Order	bigEndian	...
Ignore Case	no	
Fill Byte	0	
<b>Content</b>		
Length Kind	delimited	...
<b>Occurrences</b>		
Min Occurs <input type="text"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

34. Click on the "CompanyName" element, and in the "Delimiter" section of the Representation properties view, enter "compName=" as the Initiator and "%CR;%LF;" as the Terminator:

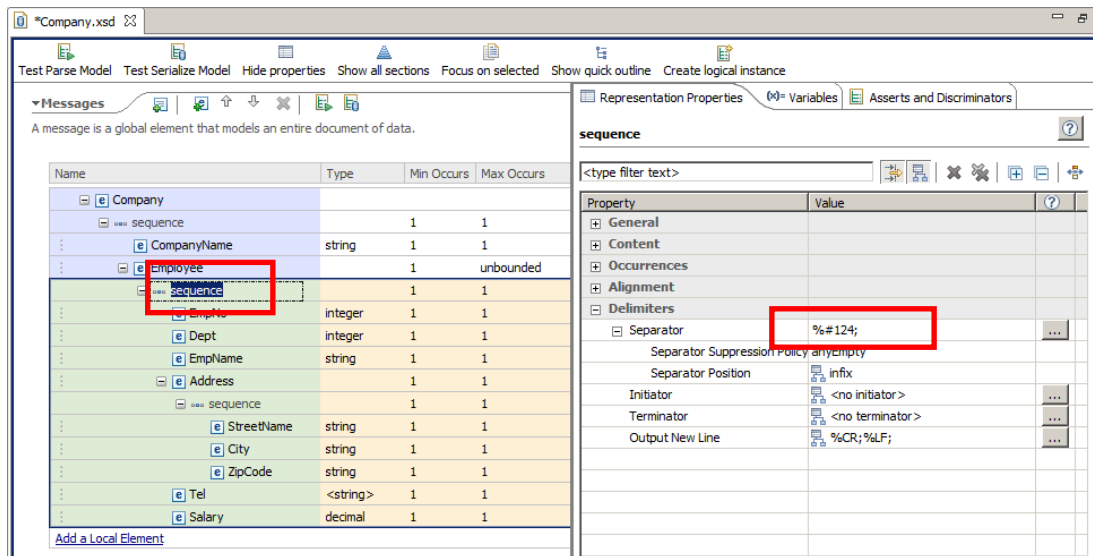


35. Click on the "Employee" element, and in the "Delimiter" section of the Representation properties view, set the Terminator value to ")%CR;%LF;". Make sure you don't miss the ")" at the start of the terminator string.

Make sure the initiator is set to "Employee(", it should have been completed automatically by the wizard at the beginning.



- Now click on the <sequence> content of the Employee element and in the Representation Properties view, check that the Separator is set to "%#124;" (the wizard should have completed it).

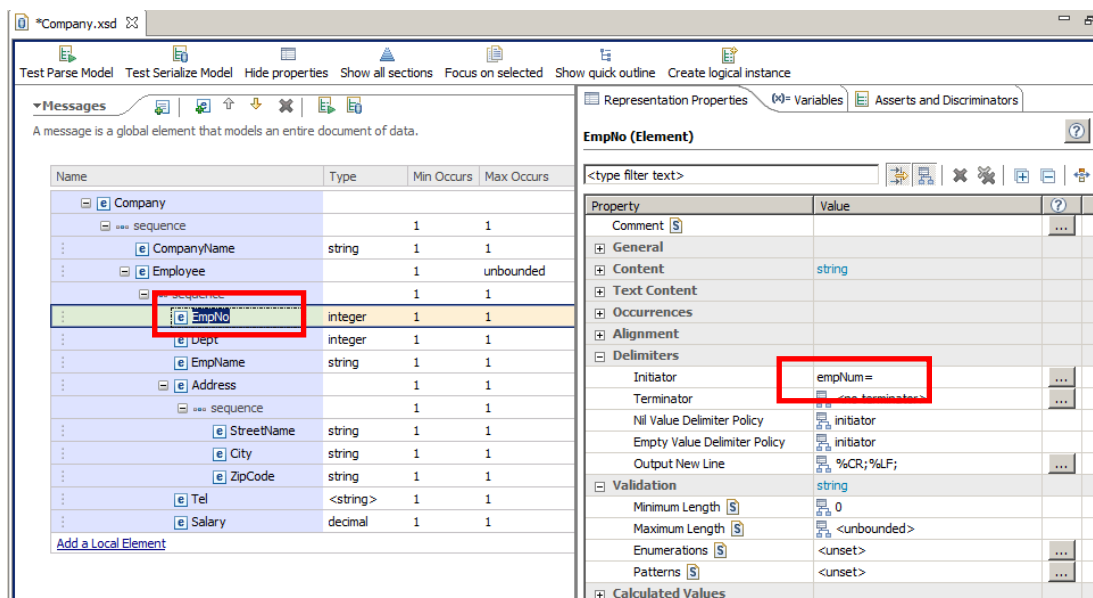


This Separator defines that all the fields inside the "Employee" structure are separated by the "|" character.

- For the fields in the Employee structure, change the Initiator of each one to the following:

Element	Initiator
EmpNo	empNum=
Dept	dept=
EmpName	empName=
Address	Addr:
Tel	tel=
Salary	sal=

Note the Address initiator uses a colon, not an "equals".





38. Save your DFDL Schema by pressing Ctrl+S or File->Save. When saved, the DFDL Schema is validated and if any errors (or warnings) are found, they will appear in the Problems view.

Make sure there are no errors in the Problems view.

## 4. Testing the Message Model

Now that the message model is complete, you can test parse it against a sample data file.

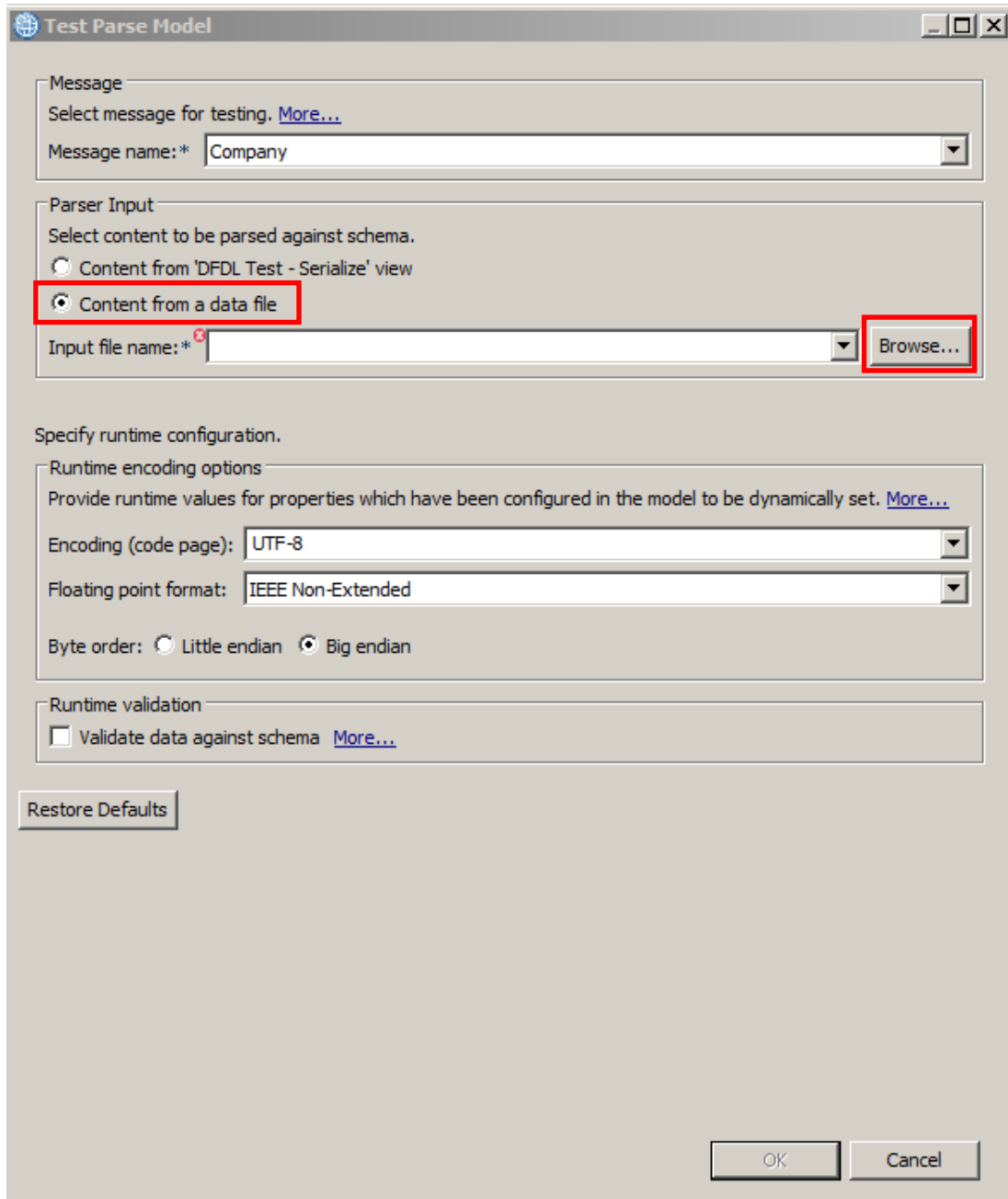
Click on the "Test Parse Model".

The screenshot shows the IBM Integration Bus V10 interface. The 'Test Parse Model' button is highlighted with a red box. Below the toolbar, the 'Messages' section is expanded, showing a table of message elements for 'Company.xsd'. The table has columns for Name, Type, Min Occurs, Max Occurs, Default Value, and Sample Value. The 'EmpNo' element is highlighted in green.

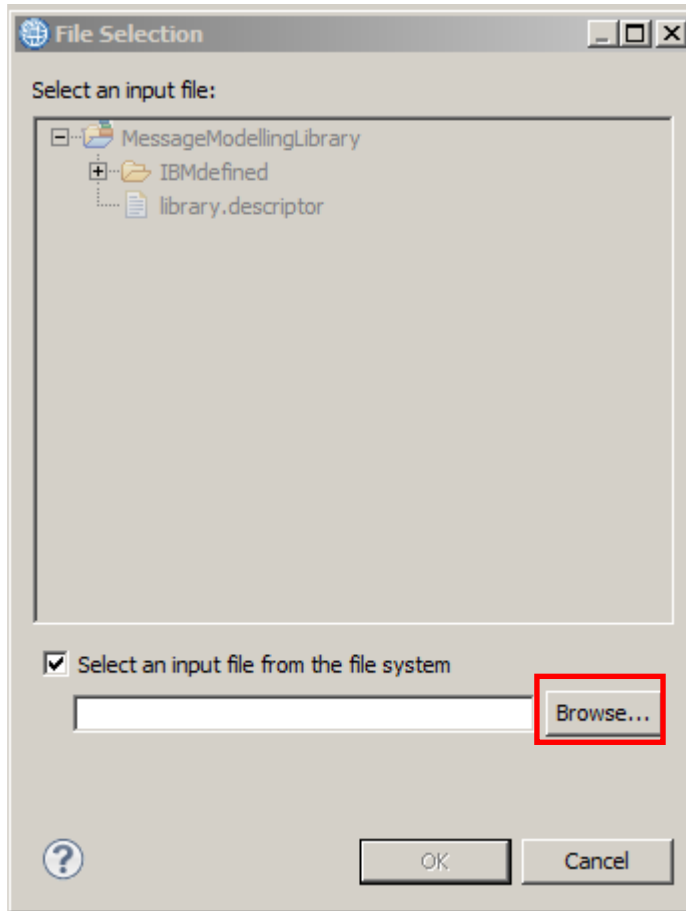
Name	Type	Min Occurs	Max Occurs	Default Value	Sample Value
[-] Company					
[-] sequence		1	1		
⋮					
[e] CompanyName	string	1	1		a
⋮					
[e] Employee		1	unbounded		
[-] sequence		1	1		
⋮					
[e] EmpNo	integer	1	1		1
⋮					
[e] Dept	integer	1	1		1
⋮					
[e] EmpName	string	1	1		a
⋮					
[+] Address		1	1		
⋮					
[e] Tel	<string>	1	1	999-999-9999	body_value5
⋮					
[e] Salary	decimal	1	1		1.0

Add a Local Element

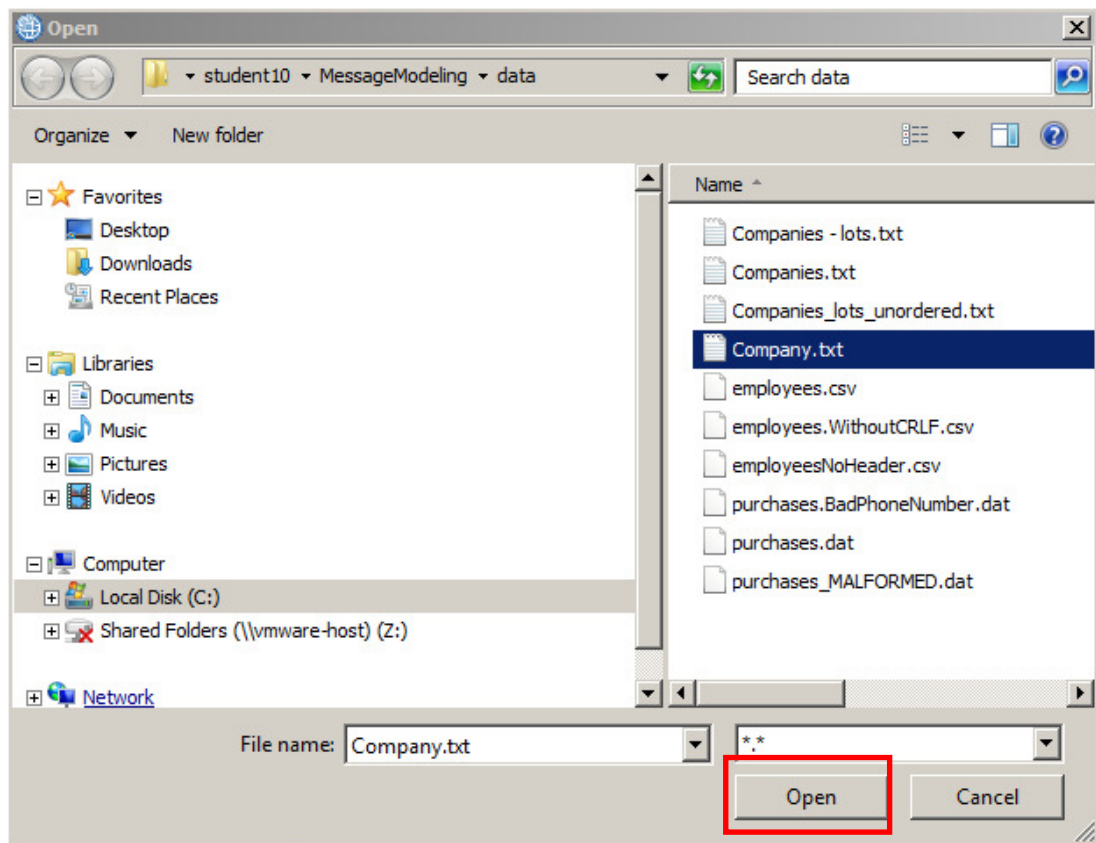
2. Select the "Content from a data file" option. Click the Browse button.



3. Check the "Select an input from the file system" checkbox, and click the Browse button.



4. Browse to the "C:\student10\MessageModeling\data" directory and select "Company.txt". Click Open, and then OK.



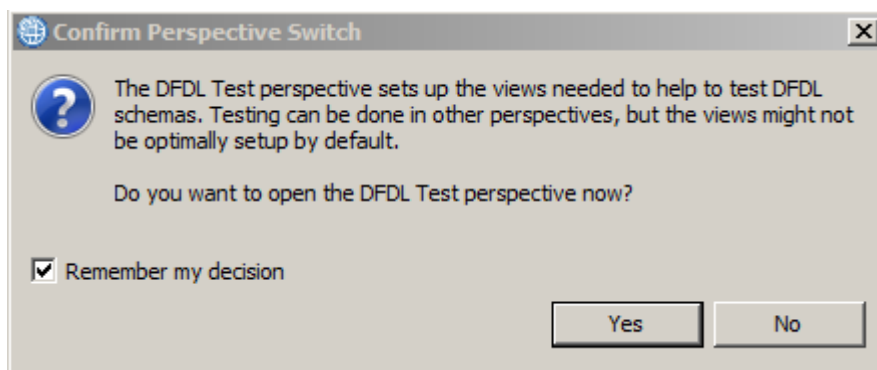
5. Check "Validate against schema" to enable validation (to test the telephone pattern you defined for validation of the "Tel" field).

Click OK.

The screenshot shows the 'Test Parse Model' dialog box with the following settings:

- Message:** Select message for testing. [More...](#)  
Message name: \* Company
- Parser Input:** Select content to be parsed against schema.
  - Content from 'DFDL Test - Serialize' view
  - Content from a data fileInput file name: \* C:\student10\MessageModeling\data\Company.txt [Browse...](#)
- Specify runtime configuration:**
  - Runtime encoding options:** Provide runtime values for properties which have been configured in the model to be dynamically set. [More...](#)
    - Encoding (code page): UTF-8
    - Floating point format: IEEE Non-Extended
    - Byte order:  Little endian  Big endian
  - Runtime validation:**  Validate data against schema [More...](#)
- [Restore Defaults](#)
- Buttons:** [OK](#) [Cancel](#)

6. If the "Confirm Perspective Switch" window appears, check the "Remember my decision" checkbox and click Yes.



7. In the "DFDL Test" perspective, "DFDL Test - Parse" view, a message bubble appears indicating the parsing was successful.

**Data source:** <From 'DFDL Test - Parse' view>  
**Message:** Company (/Users/jbadmin/IBM/IBT10/workspace/MessageMk)

Name	Type	Value
Company		
CompanyName	xs:string	My Company
Employee		
Employee		
Employee		
Employee		
Employee		

**Tips:**

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

**Parsed Input**

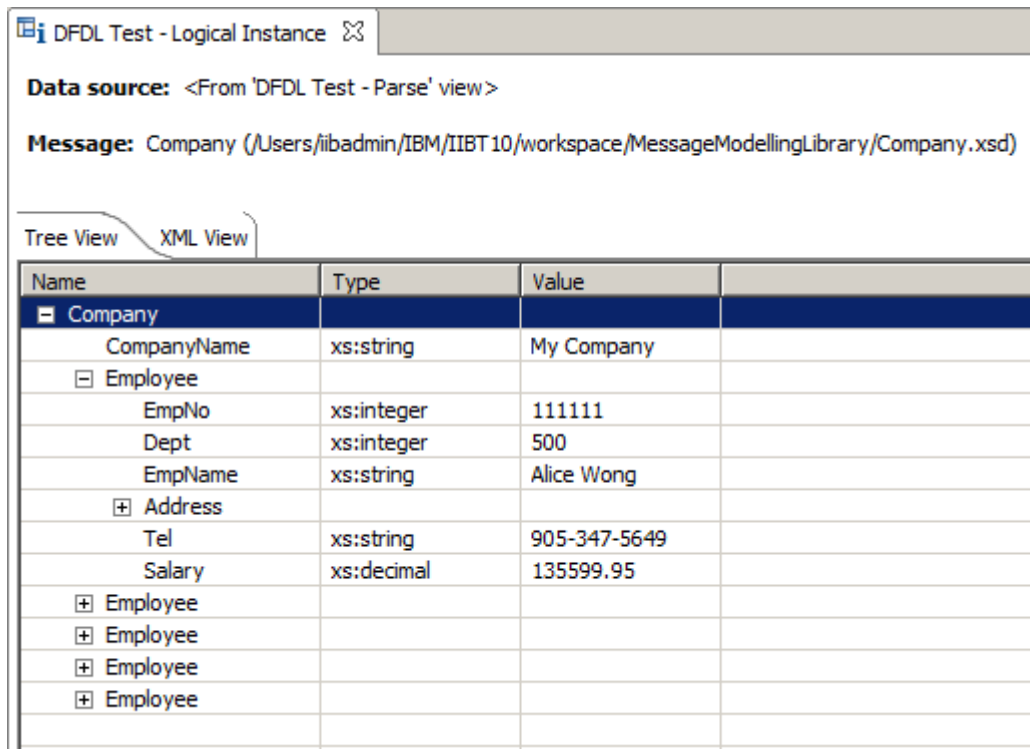
```

1 Company [compName=My Company
2 Employee (empNum=111111|dept=500|empName=Alice Wong|Addr=8200 Warden Ave,"Markham, Ont","L3G 1H7|tel=90
3 Employee (empNum=222222|dept=500|empName=James May|Addr=23 The Cuttings,Chatham,CH2 2PR|tel=208-203-13

```

Close the message by clicking on the "X".

8. Go to the "DFDL Test - Logical Instance" view, and take a look at the parsed message tree and check if it is correct.



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

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

Tree View XML View

Name	Type	Value	
[-] Company			
CompanyName	xs:string	My Company	
[-] Employee			
EmpNo	xs:integer	111111	
Dept	xs:integer	500	
EmpName	xs:string	Alice Wong	
[+] Address			
Tel	xs:string	905-347-5649	
Salary	xs:decimal	135599.95	
[+] Employee			
[+] Employee			
[+] Employee			
[+] Employee			

END OF LAB GUIDE