

# IBM Integration Bus

## Message Modeling with DFDL

### Lab 1 Modeling CSV Files

June, 2013

Hands-on lab built at product  
code level Version 9.0.0.0

# 1. Introduction to Message Modelling

A message model is used by IBM Integration Bus to model a message format. The message models used by IBM Integration Bus are all based on World Wide Web Consortium (W3C) XML Schema 1.0 (XSD).

XML Schema is an international standard that defines a language for describing the structure of XML documents. It is suited to describing the messages that flow between business applications, and it is widely used in the business community for this purpose. IBM Integration Bus uses models that are based on XML Schema to describe the structure of all kinds of message format, including message formats that are not XML.

Data Format Description Language 1.0 (DFDL) is an open standard modeling language from the Open Grid Forum (OGF) that builds upon the features of XML Schema 1.0 in order to model and validate all kinds of general text and binary data. It uses standard XSD model objects to describe the logical structure of data, together with DFDL annotations that describe the physical text or binary representation of data. IBM Integration Bus uses DFDL schema files to describe text and binary data, including industry standard formats.

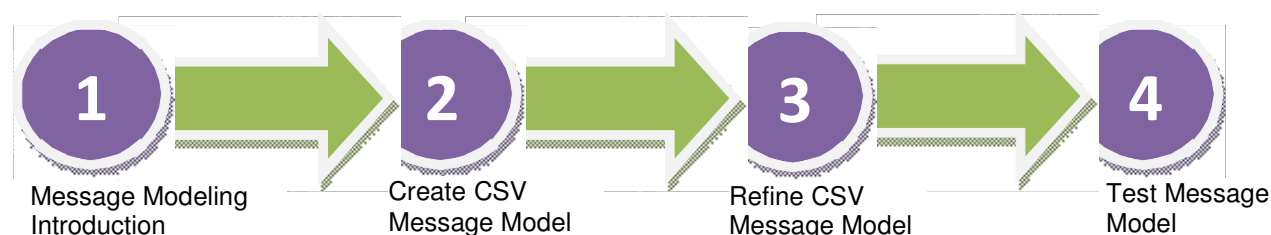
DFDL is not intended to be used to model XML documents. Use normal XML Schema files to model XML.

Support for DFDL in IBM Integration Bus includes:

- DFDL parser and domain.
- DFDL schema file creation wizards.
- DFDL schema editor for modeling text and binary data formats.
- DFDL Test perspective for testing your DFDL schema files.

For more information about DFDL you can go to the [Open Grid Forum \(OGF\)](#) web site. IBM Integration Bus supports DFDL 1.0, as defined in the following document: [Data Format Description Language \(DFDL\) 1.0](#).

Unlike previous versions, you don't need to create a Message Set project and a Message Set to model a message type (although MRM Message Models are still supported), you just have to create a DFDL schema file.



## 1.1 Lab preparation

In the pre-built vmware, all this section has been done for you.

To run this lab, unzip the supplied file MessageModelling.zip into a directory c:\student directory. This will create a subdirectory called \MessageModeling with several further subdirectories.

## 2. Creating a CSV Message Model

This lab will create a Message Model that will model a CSV file. The file has a header record, and several detail records, but no trailer record.

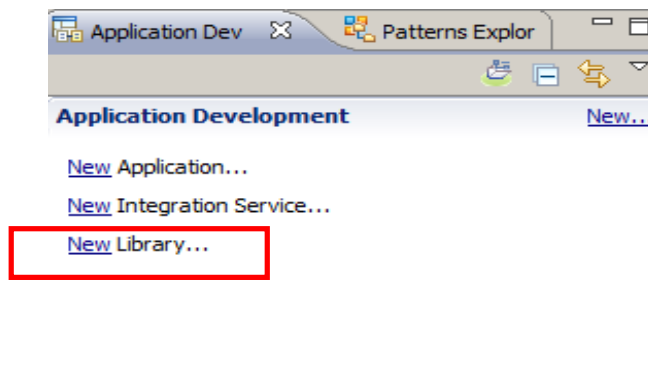
Records are delimited by CRLF characters, and fields within each record are delimited by a comma.

We will use the new Library function that was introduced in WebSphere Message Broker Version 8.

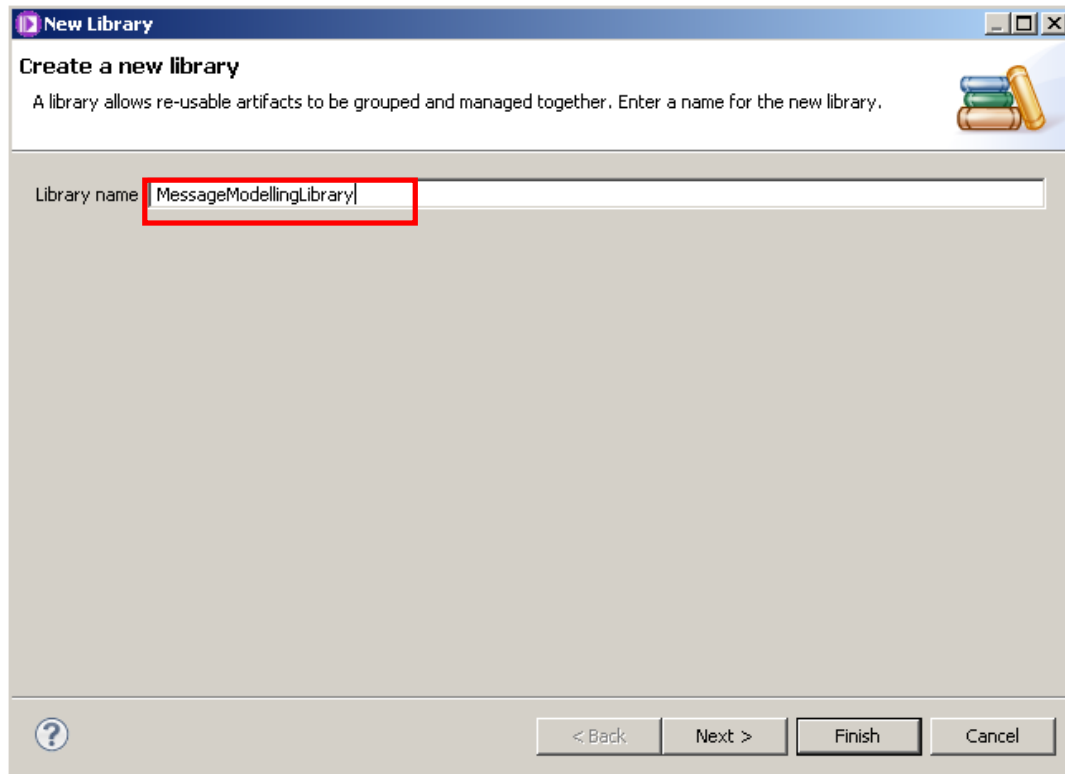
1. If not already started, start the Integration Toolkit by clicking its icon in the Start menu, or on the desktop.

Accept the default workspace location (C:\workspaces\IBWorkshop).

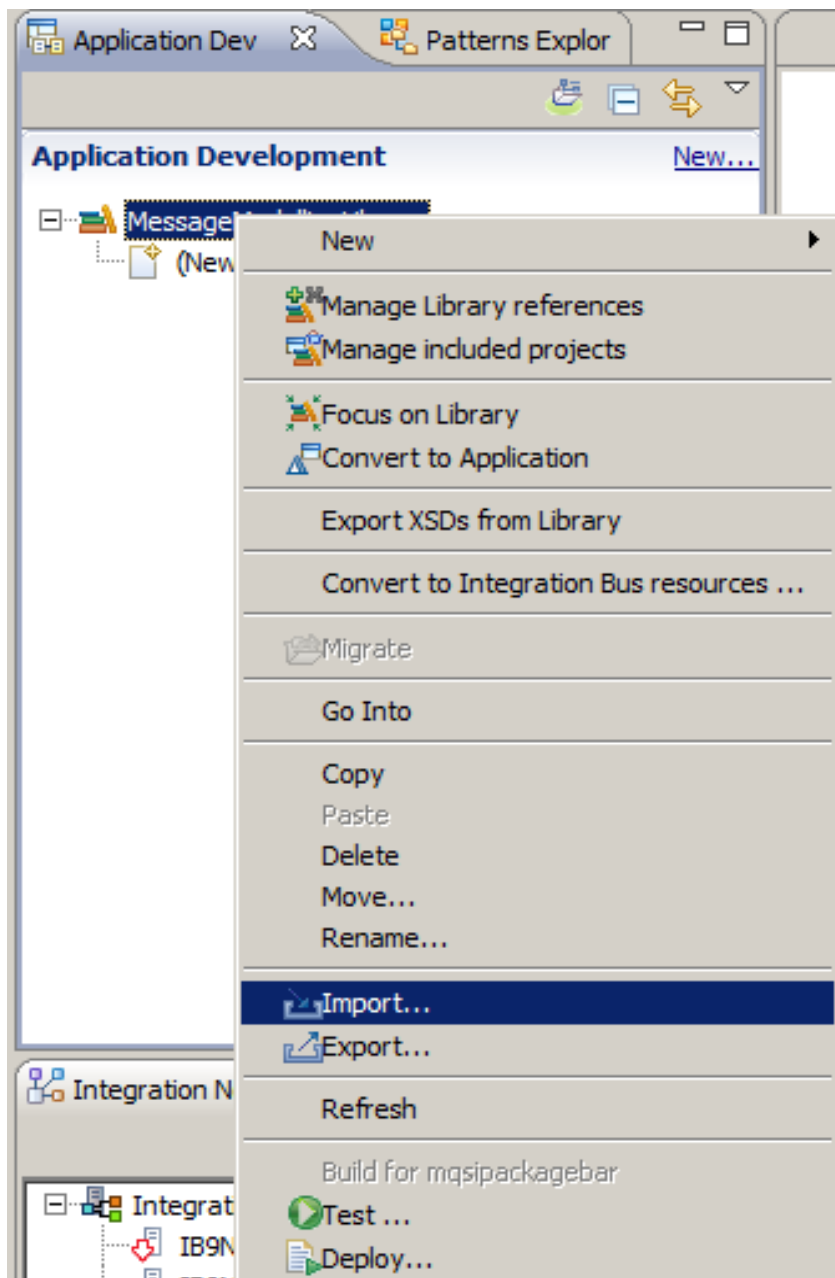
2. Create a new Library by clicking on "New library ...."



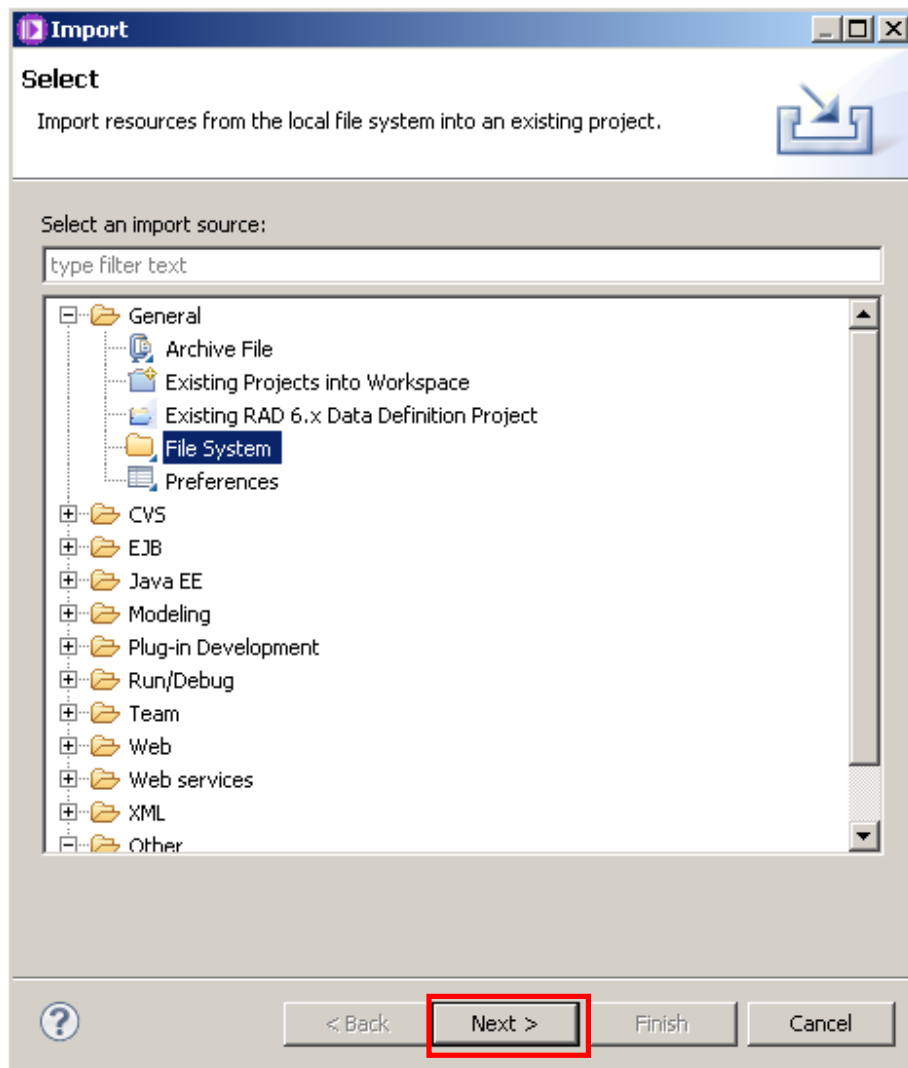
3. Call the library "MessageModellingLibrary", and click Finish.



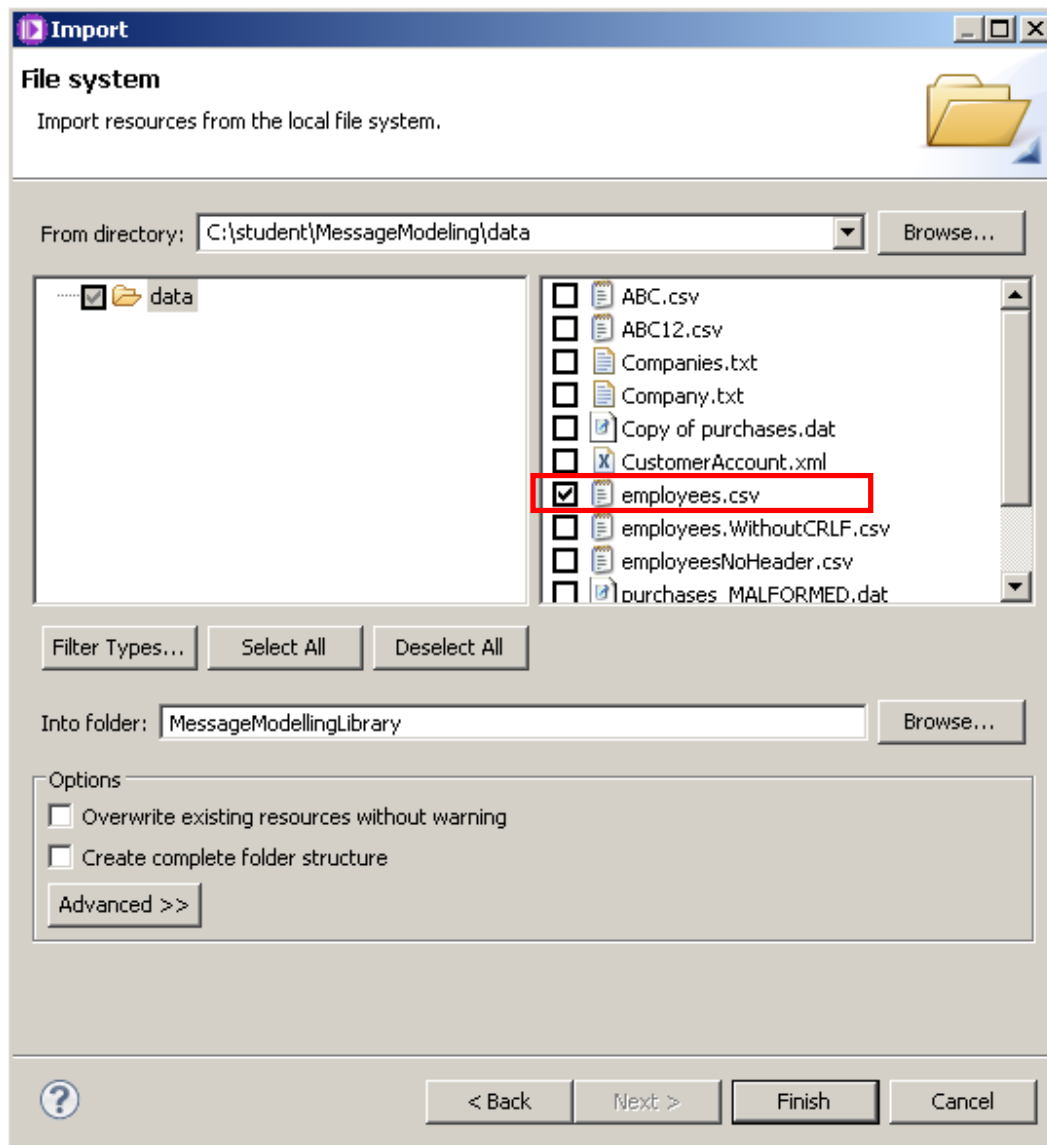
4. Now import the CSV file by right-clicking the generated Integration Bus Project's name (MessageModelingLibrary) and selecting Import.



5. In the Import window, select General -> File System and click Next.

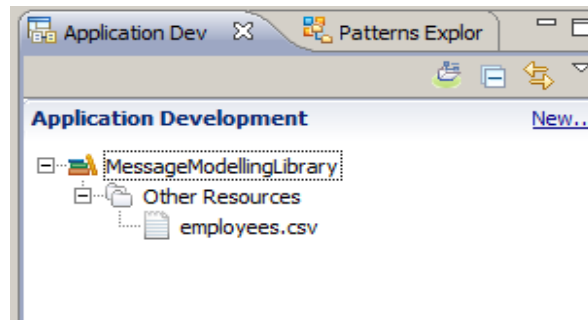


- Click Browse and select "c:\student\MessageModelling\data". In the right side of the window, select "employees.csv" and click Finish.





7. Your project should now look like this:



8. Double-click on "employees.csv" to open it, and take a quick look at it:

 A screenshot of a Notepad window titled "employees.csv - Notepad". The window displays a CSV file with a header record and 42 detail records. Each record contains 12 fields: Company, Name, ID, Department, Hire Date, Job Title, Salary, Last Name, First Name, Middle Initial, Gender, and Birth Date. The first record is:
 

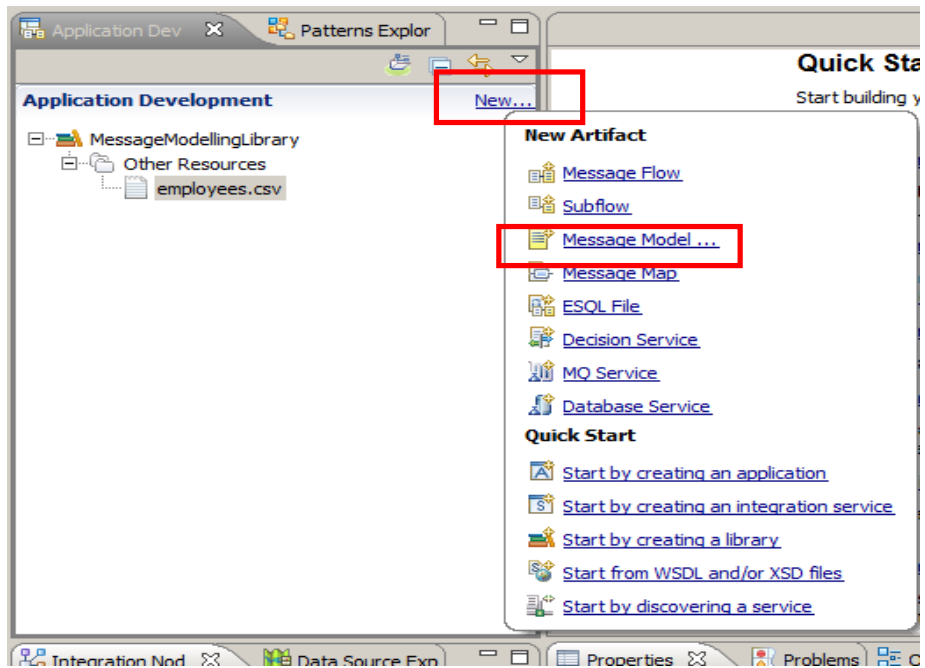
```
Company1,42,123 Street,winchester,UK
"000010","CHRISTINE","I","HAAS","A00","3978","1995-01-01","PRES","18","F","1963-08-24",152750.00
```

 The last record is:
 

```
"200140","KIM","N","NATZ","C01","1793","2006-12-15","ANALYST","18","F","1976-01-19",68420.00
```

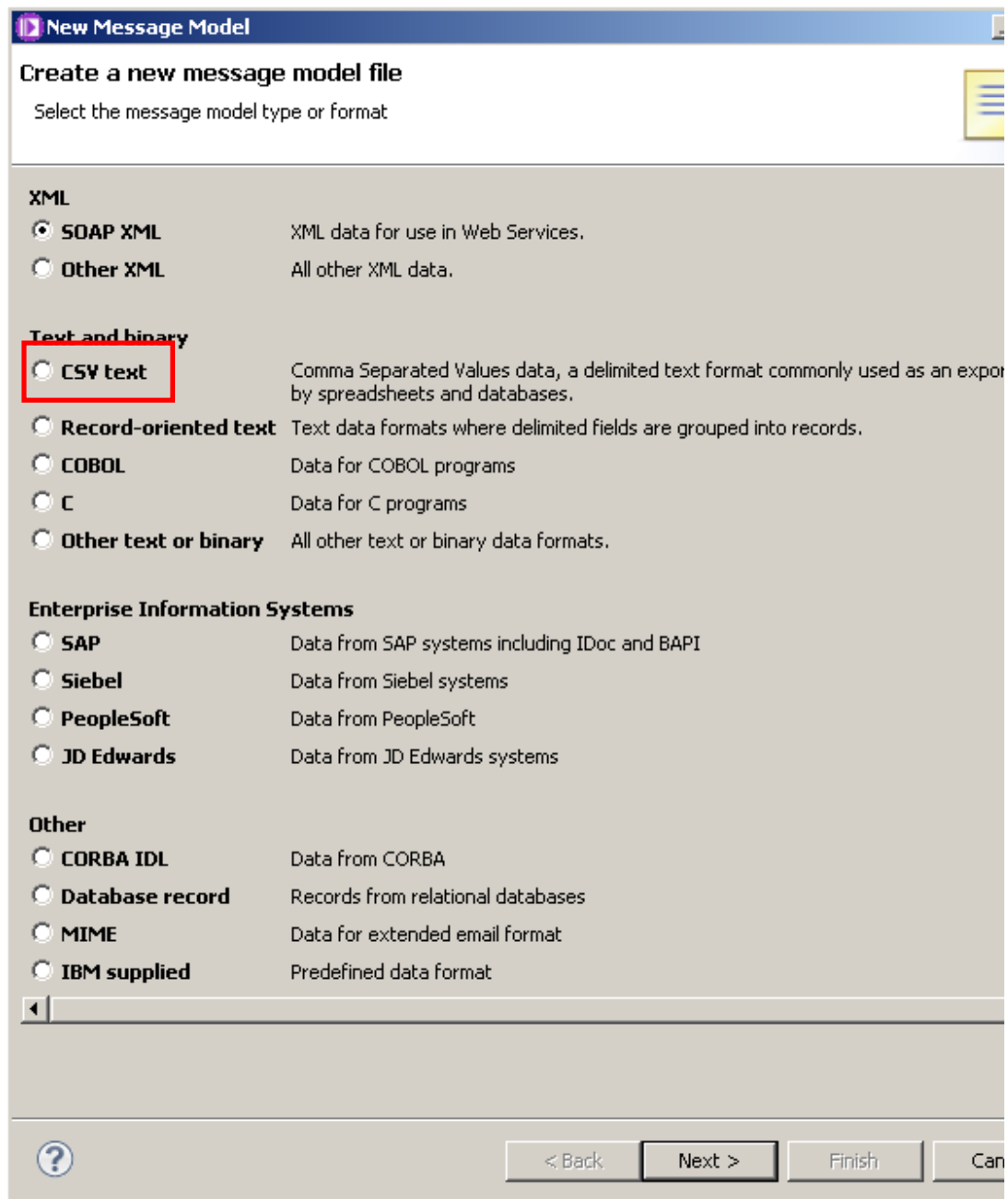
It has a header record, and 42 detail records. Each detail record has 12 fields.

9. Now we are going to create the message model. Click **New...** and select **Message Model**.



10. This will open up the Message Model creation wizard.

Select "CSV text", and click Next.



11. Then you can choose to use the CSV Message Model Creation wizard or use the DFDL editor. Select "Create a DFDL Schema file using the wizard to guide you" and click Next.

**New Message Model**

**CSV text**

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

WebSphere Message Broker requires a message model in order to parse, serialize and validate CSV data. A message model also speeds up development of your message broker applications by enabling ESQL content assist and graphical maps.

Comma Separated Values data is modeled by Data Format Description Language (DFDL) schema files. DFDL is a standard from the Open Grid Forum for describing all kinds of text and binary data.

Create a DFDL schema file using this wizard to guide you.

Create an empty DFDL schema file, I will model my data using the DFDL schema editor

Import or replace the IBM supplied DFDL schema property defaults for CSV.

	A	B	C	D	E
1	Year	Make	Model	Description	Price
2	2009	SK Inc	MBTk7	4293cc, V8	53880.00
3	2010	Hans On	DFDL	3000cc straight 6	31395.00
4	2010	AOD corp	MB8	4163cc, V8	51435.00

Export

Year,Make,Model,Description,Price  
2008,SK Inc,MBTk7,"4293cc, V8",53880.00  
2010,Hans On,DFDL,3000cc straight 6,31395.00  
2010,AOD corp,MB8,"4163cc, V8",51435.00

< Back   Next >   Finish   Cancel

- The Project name will have already been filled in for you. (If it hasn't, click browse, and select the MessageModellingLibrary).

Enter "Employee" as the name for the DFDL Schema file 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:

13. In the next screen the wizard lets us choose some settings about the CSV format, like the End of Record character, Blank Records treatment, Number of fields, Encoding and Escape Scheme.

Leave the End of Record character as "Carriage Return & Line Feed".  
Check "The first record is a header".  
Set the number of fields to 12.

Click Finish to complete the wizard.

**New Message Model**

### Configure schema for CSV data

Provide settings for a new schema that will model CSV data.

**Record settings**

End of record character: Carriage Return & Line Feed - %CR;%LF;

(Blank records will be skipped)

The first record is a header

**Field settings**

Number of fields: 12

Create default values for fields

**Encoding code page options:**

Dynamic (provided to the processor by the application at runtime)

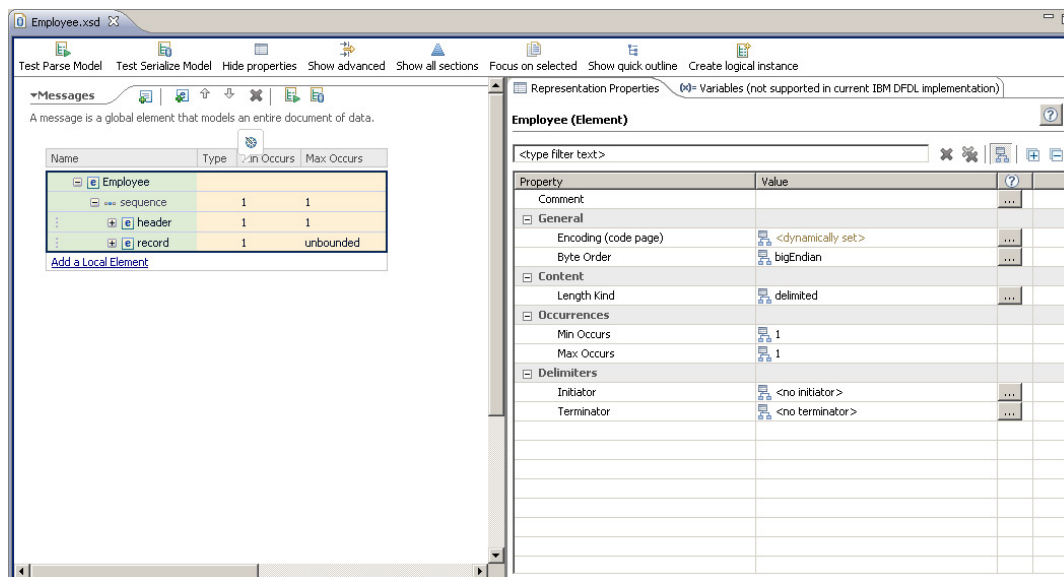
Fixed UTF-8

**Global settings**

Escape scheme: CSV Escape Scheme

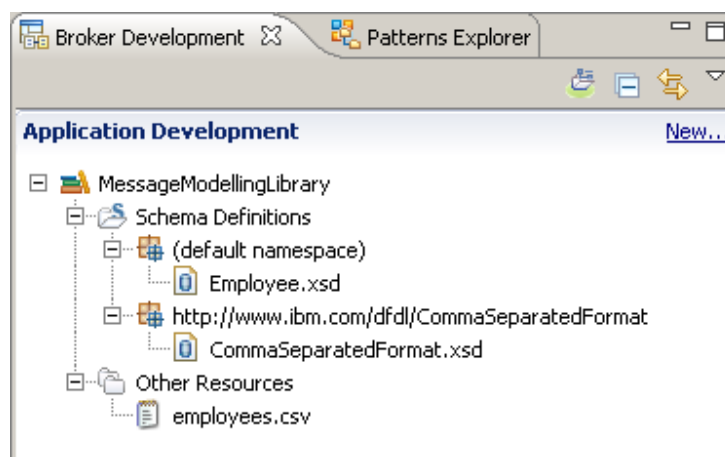
? < Back Next > Finish Cancel

14. Now the DFDL editor should show the CSV message model.



Notice that there's a new folder in our project in the default namespace, called "Schema definitions" with an XSD file called Employee.xsd (under the default namespace). This is the message model we've just created.

Note that this is a standard DFDL XSD, with no specific annotations for Integration Bus.

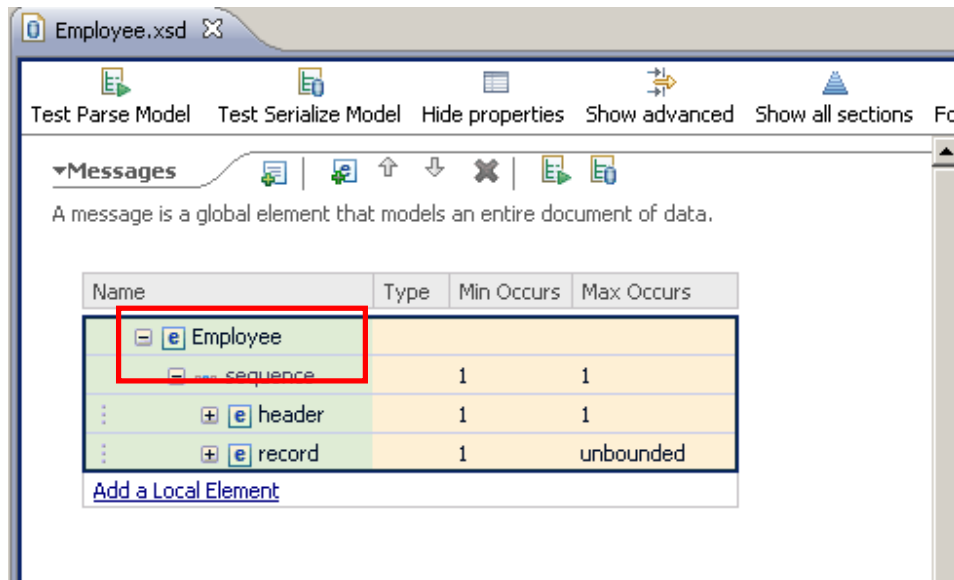


You'll also notice that there is another XSD file under Schema Definitions called "CommaSeparatedformat.xsd". This "Helper schema file" was automatically added by the CSV wizard and contains CSV-specific defaults for all the DFDL properties. This is required because DFDL doesn't have built-in defaults, so if an object needs a property, a value must be supplied. To ease this task, the wizard creates a helper DFDL schema for each kind of data (COBOL, CSV, etc.)

These files are related by an import statement in the schema references section of the Employee.xsd file.

15. You can differentiate which properties have an "inherited" value that came from the Helper schema from those defined by you, by detecting the presence of the "inheritance" icon.

In the Message Roots view (the left-hand side of the Message Model editor) click on the Employee element.

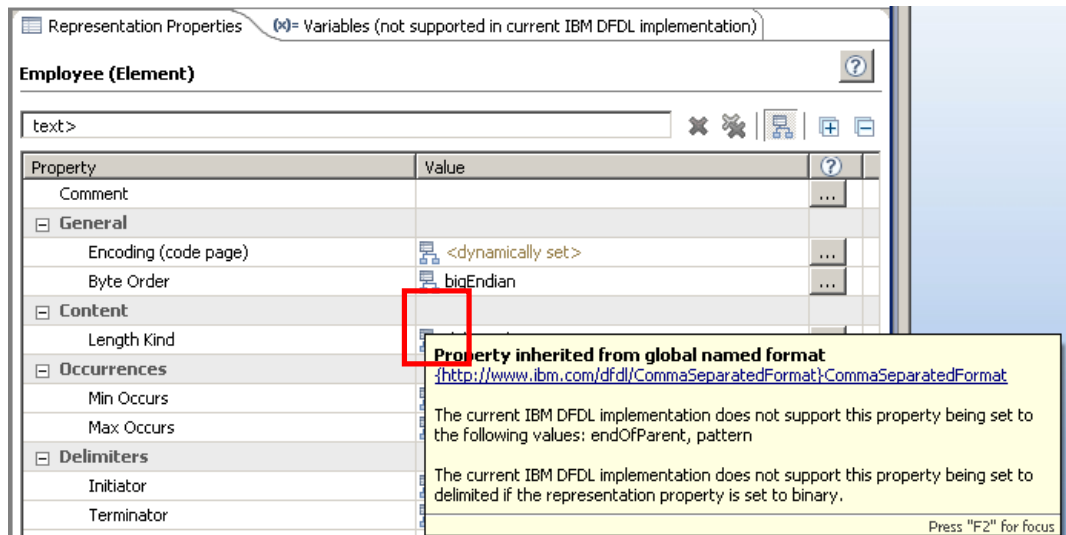




16. Take a look at the Representation Properties view on the right hand side. Notice the "inheritance" icon to the left of the "Length Kind" property.

Hover over the Inheritance icon to discover its origin.

In this case, the "Length Kind" property was inherited from the "CommaSeparatedFormat" Helper schema that the wizard automatically imported.



The screenshot shows the 'Representation Properties' view for an 'Employee (Element)'. The 'Length Kind' property is highlighted with a red box. A tooltip is displayed over the inheritance icon, indicating that the property is inherited from a global named format.

Property	Value
Comment	
<b>General</b>	
Encoding (code page)	<dynamically set>
Byte Order	bigEndian
<b>Content</b>	
Length Kind	
<b>Occurrences</b>	
Min Occurs	
Max Occurs	
<b>Delimiters</b>	
Initiator	
Terminator	

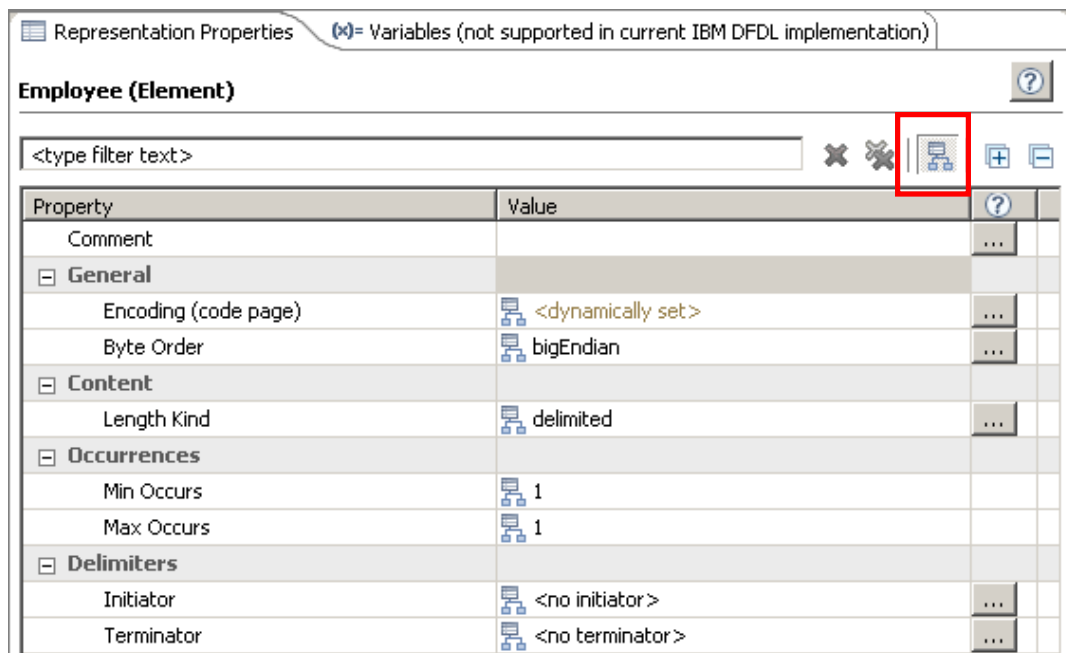
**Property inherited from global named format**  
{<http://www.ibm.com/dfdl/CommaSeparatedFormat>}-CommaSeparatedFormat

The current IBM DFDL implementation does not support this property being set to the following values: endOfParent, pattern

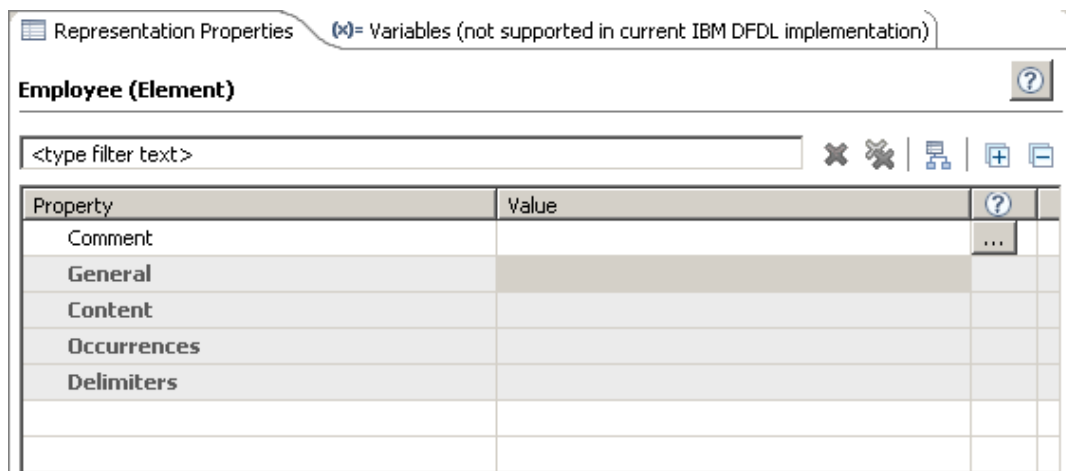
The current IBM DFDL implementation does not support this property being set to delimited if the representation property is set to binary.

Press "F2" for focus

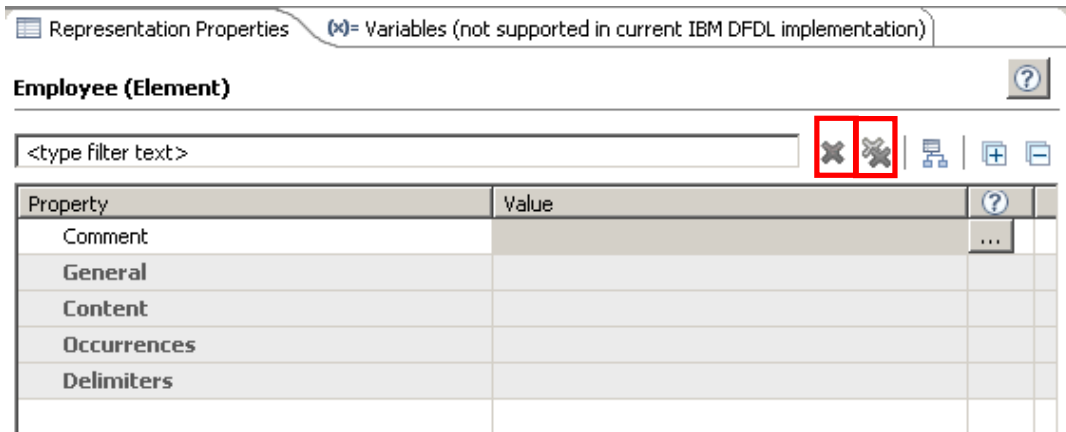
17. Click on the "Show Local Properties" icon.



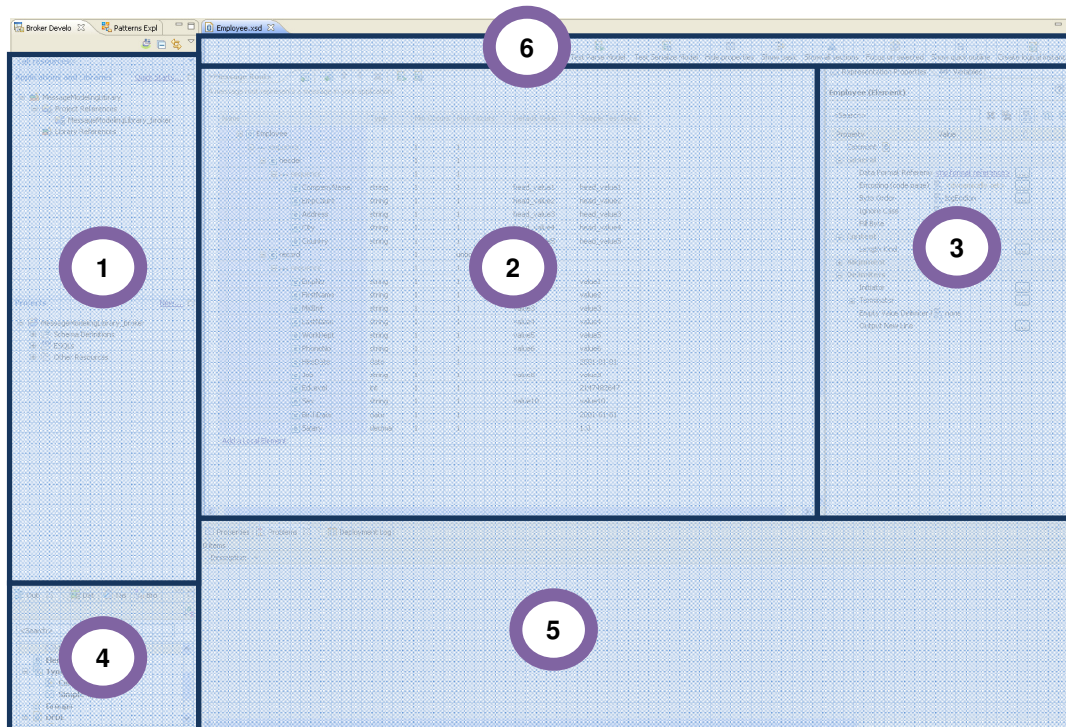
18. Notice that now all the inherited properties are hidden. Only the locally defined properties are shown. (in this case there are none)



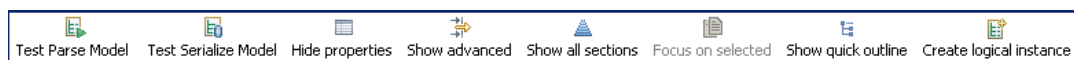
- The other two icons to the left of the "Show Local Properties"/ "Show Inherited Properties" icon, allow you to clear a selected local property or all of them and replace them with the inherited value.



20. Take a look at the different parts of the DFDL Editor:



- 1) **Applications and Libraries / Projects view:** In the top of this view you'll find the Applications/Libraries, and in the bottom the projects. All the DFDL XSD files will be located under the "Schema Files" folder inside of a project.
- 2) **Logical structure view:** This is the core of the DFDL Editor. Here you will work with the Message Models.
- 3) **DFDL properties:** Shows the DFDL properties of the selected elements in the editor view. These were moved from the traditional properties window to provide better viewing.
- 4) **Outline view:** Shows the complete outline of the current DFDL Message Model.
- 5) **Problems view:** In this view you will be able to see all the warning and errors of your project.
- 6) **Icon Bar:** Several actions can be started from the icons in this bar:



- **Test Parse Model:** Launches the DFDL Test Perspective to test-parse sample data against your selected message model.
- **Test Serialize Model:** Launches the DFDL Test Perspective, to test-serialize sample data by using the selected DFDL schema.
- **Hide properties:** Hides the Representation Properties view.
- **Show advanced:** Shows the advanced Representation Properties.
- **Show all sections:** Shows all the available sections of the DFDL Message Model in the Editor.
- **Focus on selected:** Shows only the selected element in the Editor.
- **Show quick outline:** Shows a hanging outline view of the message model in the Editor.

**Create logical instance:** using default values and sample test data, specified in this DFDL file

21. The DFDL Editor in the center of the screen describes data elements.

Name	Type	Min Occurs	Max Occurs	Default Value	Sample Value
[-] [e] Employee					
[-] ... sequence		1	1		
⋮					
[+] [e] header		1	1		
⋮					
[-] [e] record		1	unbounded		
[-] ... sequence		1	1		
⋮					
[e] field1	string	1	1		value1
[e] field2	string	1	1		value2
[e] field3	string	1	1		value3
[e] field4	string	1	1		value4
[e] field5	string	1	1		value5
[e] field6	string	1	1		value6
[e] field7	string	1	1		value7
[e] field8	string	1	1		value8
[e] field9	string	1	1		value9
[e] field10	string	1	1		value10
[e] field11	string	1	1		value11
[e] field12	string	1	1		value12
<a href="#">Add a Local Element</a>					

Take a look at the columns:

- Name: name of the data element
- Type: data type (string, int, boolean, decimal, date, etc)
- Min Occurs: minimum amount of occurrences expected (0 or greater)
- Max Occurs: maximum amount of occurrences expected (from 1 to unbounded)
- Default value: you can specify a default value for each field
- Sample value: The wizard has created a sample value based on the element type

### 3. Refine the Generated Message Model

A number of refinements need to be made to the model that the wizard has created:

1. Adjust the number of elements in the header.
2. Change the names of the field elements to reflect the field usage, and adjust the data types of some the elements.
3. Remove the generated value of the Separator field for the Employee sequence.
4. On the header definition, set the Terminator property to CRLF.
5. On the record definition, set the Terminator property to CRLF.
6. Remove all default values created by the wizard (a current limitation for CSV message models).
7. Configure the model to permit the final character to be missing.

The change from the use of Separators to Terminators is necessary because the CSV wizard generates the model with Separator 'CR/LF' and Separator Position 'Infix'. This models data where there is no final trailing CR/LF. If there is always a final trailing CR/LF then changing the Separator Position to 'Postfix' is needed.

If you are modelling a case where some instances of the data have no final trailing CR/LF, but other instances do (as we assume in this case), then the delimiter needs to be modelled as a Terminator, with the special property Document Final Terminator Can Be Missing set to 'Yes'.

1. Delete fields 6 to 12 from the "header" element. Select all of them and right-click -> delete (or press the Delete key).

To select the elements, you can either select the range (select field 6, then hold the upper-case key and select element 12), or you can select multiple individual elements by using the Ctrl key to select multiple elements.

Select the elements by placing the mouse on the left side of the element, not the element name.

Name	Type	Min Occurs	Max Occurs	Default Value	Sample Value
Employee					
sequence		1	1		
header		1	1		
sequence		1	1		
head_field1	string	1	1		head_value1
head_field2	string	1	1		head_value2
head_field3	string	1	1		head_value3
head_field4	string	1	1		head_value4
head_field5	string	1	1		head_value5
head_field6	string	1	1		head_value6
head_field7	string	1	1		head_value7
head_field8	string	1	1		head_value8
head_field9	string	1	1		head_value9
head_field10	string	1	1		head_value10
head_field11	string	1	1		head_value11
head_field12	string	1	1		head_value12
separator		1	unbounded		

- Change the names of the remaining 5 header fields by clicking on their names to these. Use the down-arrow key to move to the element below.

Name	Type	Min Occurs	Max Occurs
[-] Employee			
[-] sequence		1	1
⋮			
[-] header		1	1
[-] sequence		1	1
⋮			
[-] <b>CompanyName</b>	string	1	1
[-] <b>EmpCount</b>	string	1	1
[-] <b>Address</b>	string	1	1
[-] <b>City</b>	string	1	1
[-] <b>Country</b>	string	1	1
⋮			
[+] record		1	unbounded
<a href="#">Add a Local Element</a>			

- Expand the "record" element to show the 12 fields created by the wizard. Change their names, to reflect the CSV field names.

The screenshot shows the Message Modeler interface for 'Employee.xsd'. The 'record' element is expanded to show a sequence of 12 fields, each of type 'string' and occurring once. The fields are named 'field1' through 'field12'. The interface includes a toolbar with options like 'Test Parse Model', 'Test Serialize Model', and 'Hide properties'. A 'Messages' pane at the top shows a message icon and a description: 'A message is a global element that models an entire document of data.'

Name	Type	Min Occurs	Max Occurs
[-] Employee			
[-] sequence		1	1
⋮			
[+] header		1	1
⋮			
[-] record		1	unbounded
[-] sequence		1	1
⋮			
[-] field1	string	1	1
[-] field2	string	1	1
[-] field3	string	1	1
[-] field4	string	1	1
[-] field5	string	1	1
[-] field6	string	1	1
[-] field7	string	1	1
[-] field8	string	1	1
[-] field9	string	1	1
[-] field10	string	1	1
[-] field11	string	1	1
[-] field12	string	1	1
<a href="#">Add a Local Element</a>			

## 4. Change the field names to the following:

- 1) EmpNo
- 2) FirstName
- 3) MidInit
- 4) LastName
- 5) WorkDept
- 6) PhoneNo
- 7) HireDate
- 8) Job
- 9) EdLevel
- 10) Sex
- 11) BirthDate
- 12) Salary

It should look like this:

Name	Type	Min Occurs	Max Occurs
[-] Employee			
[-] sequence		1	1
⋮			
[+] header		1	1
⋮			
[-] record		1	unbounded
[-] sequence		1	1
⋮			
[+] EmpNo	string	1	1
⋮			
[+] Firstname	string	1	1
⋮			
[+] MidInit	string	1	1
⋮			
[+] LastName	string	1	1
⋮			
[+] WorkDept	string	1	1
⋮			
[+] PhoneNo	string	1	1
⋮			
[+] HireDate	string	1	1
⋮			
[+] Job	string	1	1
⋮			
[+] EdLevel	string	1	1
⋮			
[+] sex	string	1	1
⋮			
[+] BirthDate	string	1	1
⋮			
[+] salary	string	1	1
<a href="#">Add a Local Element</a>			



- Now change the data type for some fields. Click in the Type column of the HireDate field and select "date" as the data type.

Name	Type	Min Occurs	Max Occurs
Employee			
sequence		1	1
header		1	1
record		1	unbounded
sequence		1	1
EmpNo	string	1	1
Firstname	string	1	1
MidInit	string	1	1
LastName	string	1	1
WorkDept	string	1	1
PhoneNo	string	1	1
HireDate	string	1	1
Job			
EdLevel			
sex			
BirthDate			
salary			

[Add a Local Element](#)

- Browse...
- <Anony...
- boolean
- byte
- date**
- dateTime
- decimal
- double
- float
- hexBinary
- int

6. Repeat this step for BirthDate, setting type to "date".

Change EdLevel's data type to "int".

Name	Type	Min Occurs	Max Occurs
Employee			
sequence		1	1
header		1	1
record		1	unbounded
sequence		1	1
EmpNo	string	1	1
Firstname	string	1	1
MidInit	string	1	1
LastName	string	1	1
WorkDept	string	1	1
PhoneNo	string	1	1
HireDate	date	1	1
Job	string	1	1
EdLevel	string	1	1
sex			
BirthDate			
salary			

float
hexBinary
<b>int</b>
integer
long
nonNegativeInteger
short
string
time
unsignedByte
unsignedInt

7. Finally, change Salary field data type to decimal. Your message model should now look like this:

Name	Type	Min Occurs	Max Occurs	Default Value	Sample Value
[-] Employee					
[-] sequence		1	1		
⋮					
[-] header		1	1		
[-] sequence		1	1		
⋮					
[-] CompanyName	string	1	1		head_value1
[-] EmpCount	string	1	1		head_value2
[-] Address	string	1	1		head_value3
[-] City	string	1	1		head_value4
[-] Country	string	1	1		head_value5
⋮					
[-] record		1	unbounded		
[-] sequence		1	1		
⋮					
[-] EmpNo	string	1	1		value1
[-] Firstname	string	1	1		value2
[-] MidInit	string	1	1		value3
[-] LastName	string	1	1		value4
[-] WorkDept	string	1	1		value5
[-] PhoneNo	string	1	1		value6
[-] HireDate	date	1	1		2010-12-31
[-] Job	string	1	1		value8
[-] EdLevel	int	1	1		1
[-] Sex	string	1	1		value10
[-] BirthDate	date	1	1		2010-12-31
[-] Salary	decimal	1	1		1.0

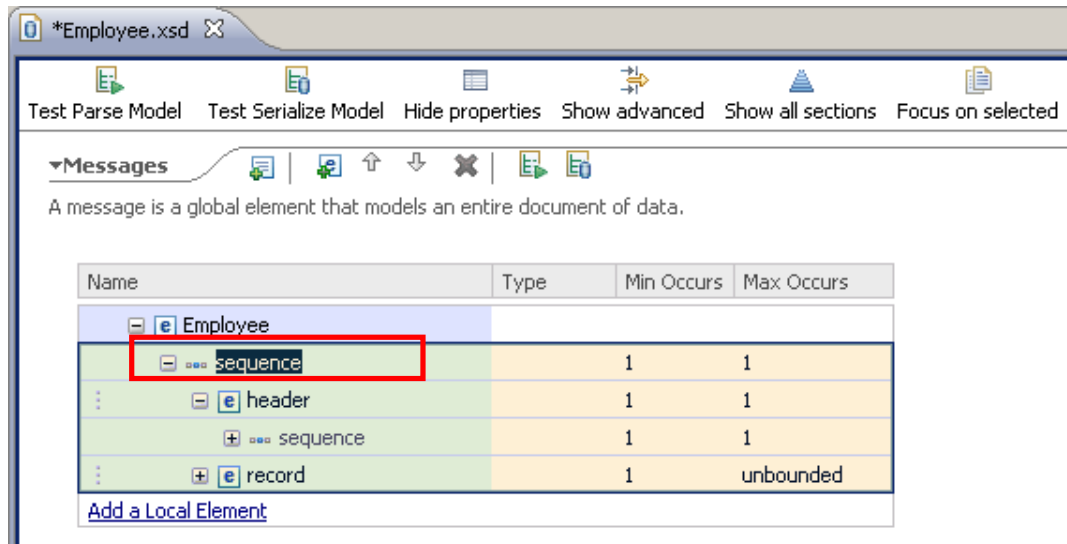
[Add a Local Element](#)

8. **IMPORTANT NOTE – WebSphere Message Broker V8 Fixpack 2 (also referred to as V8.0.0.2) has changed some default values generated by the CSV wizard. If you are using V8.0.0.2 or IBM Integration Bus V9.0, you may now skip to step 16.**

If you are using V8.0 or V8.0.0.1, you should perform steps 9 through 16.

9. In the Employee sequence, reset the generated value for the Separator.

In the DFDL editor, click on the <sequence> content of the Employee element.

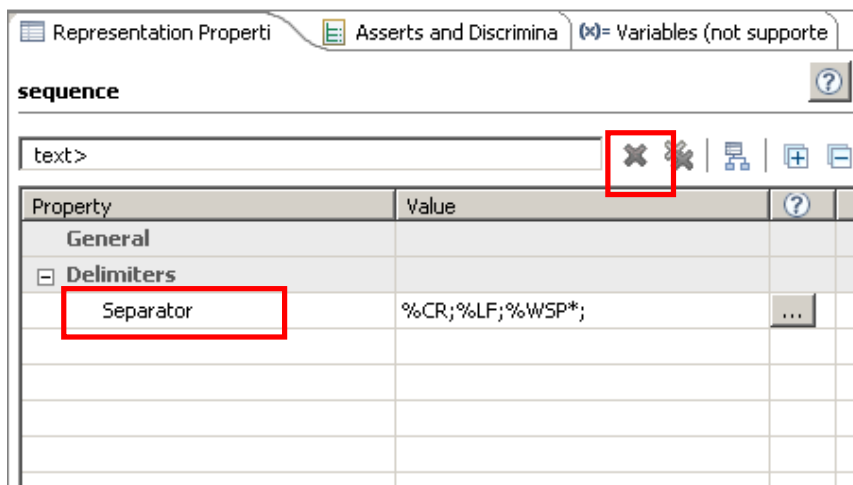


The Separator between records is set to the default value of "Carriage Return/Line Feed". Notice that there is a third Separator "%WSP\*" which was added by the wizard because it automatically skips blank records.

However, in this case, we are going to use Terminators to indicate a new record or row, so we need to remove this value.

Highlight the word "Separator", and click the single grey cross to remove the value. The new value will show as <no separator>.

(You may find that the editor initially replaces the generated value with a ",". Ensure this is completely removed).



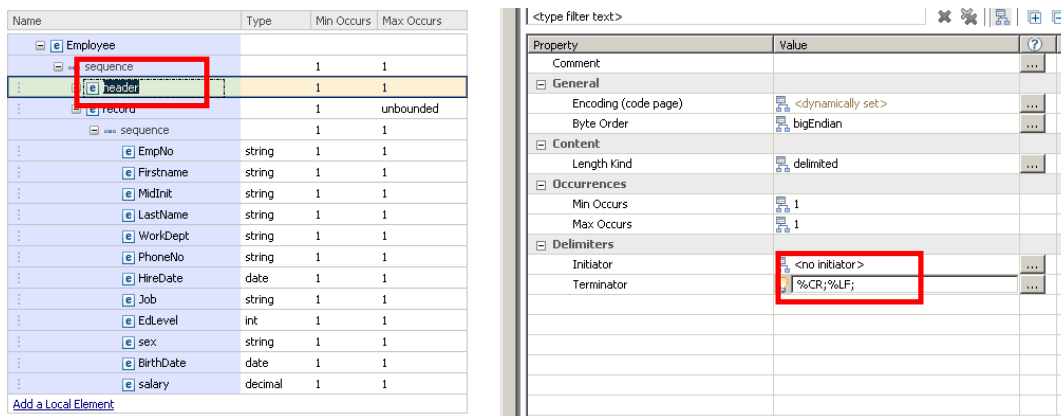
- On the “header” element, we need to specify a Terminator with value CR-LF.

Click the “header” element. In the Property pane, under Delimiters; the Terminator field will have a value of <no terminator>. Click in this field, and then use the Content Assist feature to set it to the value %CR;%LF;

Note that this lab does not need to include the %WSP value, because the test data does not have any blank lines. However, in the more general case, it is a good idea to include %WSP as well, which will mean that blank lines in the data are handled (excluded).

Content Assist is started by the key combination Ctrl-Space. Then use the down-arrow to position to the required value; select the value with the Return key.

When the field has been populated correctly, click Return to make sure the field is updated.



When complete the Terminator for the “header” element should look like this:

Delimiters	
Initiator	<no initiator>
Terminator	%CR;%LF;

- Repeat the above step for the “record” element.

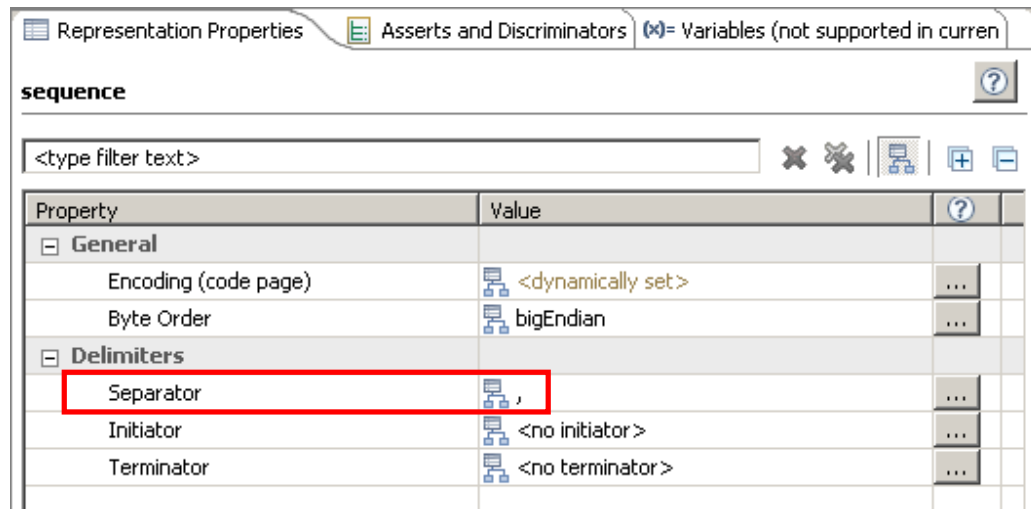
When complete, the Terminator for the “record” element should look like this:

Delimiters	
Initiator	<no initiator>
Terminator	%CR;%LF;

12. Check the values for the <sequence> content of the header element and the <sequence> content of the record element to check that the field separator is set to ",".

The Initiator and Terminator values should be <no initiator> and <no terminator>.

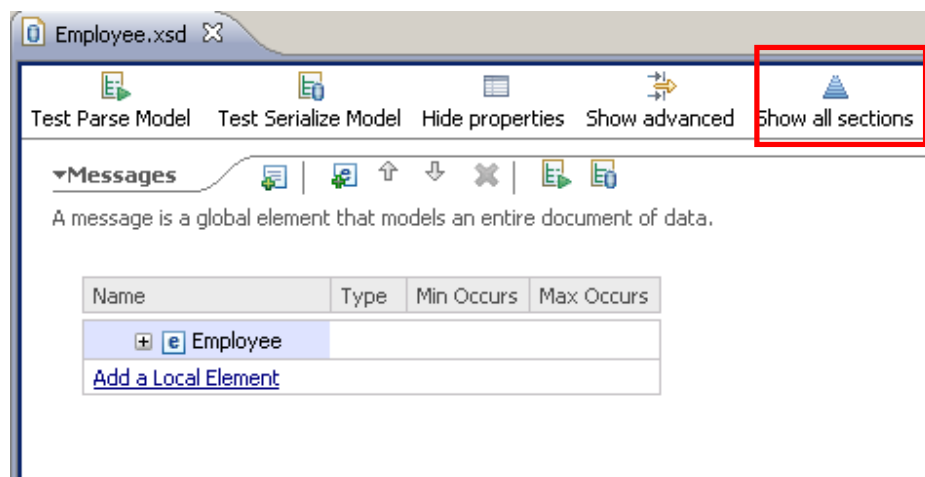
No changes should be necessary here.



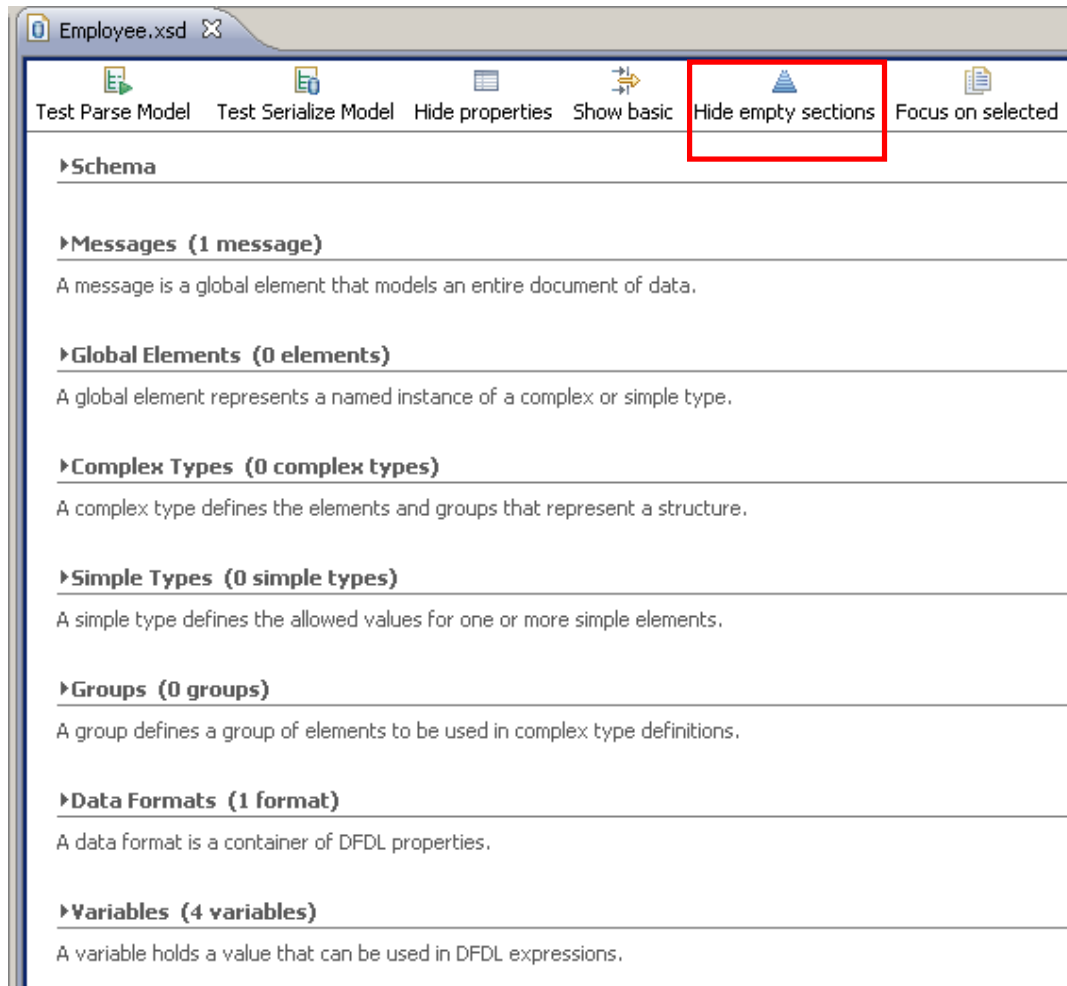
13. Finally, a CSV file may, or may not, have a Carriage Return Line Feed as the last character. In some cases, the last record may be missing the final "new line" character.

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

IN the editor, collapse the Employee model, and then click "Show all sections".



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



## 15. Expand the Data Formats section.

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

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

The screenshot shows the IBM Integration Bus Message Modeling tool interface. The left pane displays the 'Data Formats' section with a table:

Name	Type
<default format>	Definition Format

The right pane shows the 'Representation Properties' for the selected '<default format> (Data Format)'. The 'Delimiters' section is expanded, and the 'Document Final Terminator Can Be Missing' property is highlighted with a red box and set to 'yes'.

Property	Value
Separator	,
Separator Policy	suppressed
Separator Position	infix
Initiator	<no initiator>
Terminator	<no terminator>
Document Final Terminator Can Be Missing	yes
Nil Value Delimiter Policy	none
Empty Value Delimiter Policy	none
Output New Line	%CR;%LF;
Calculated Values	

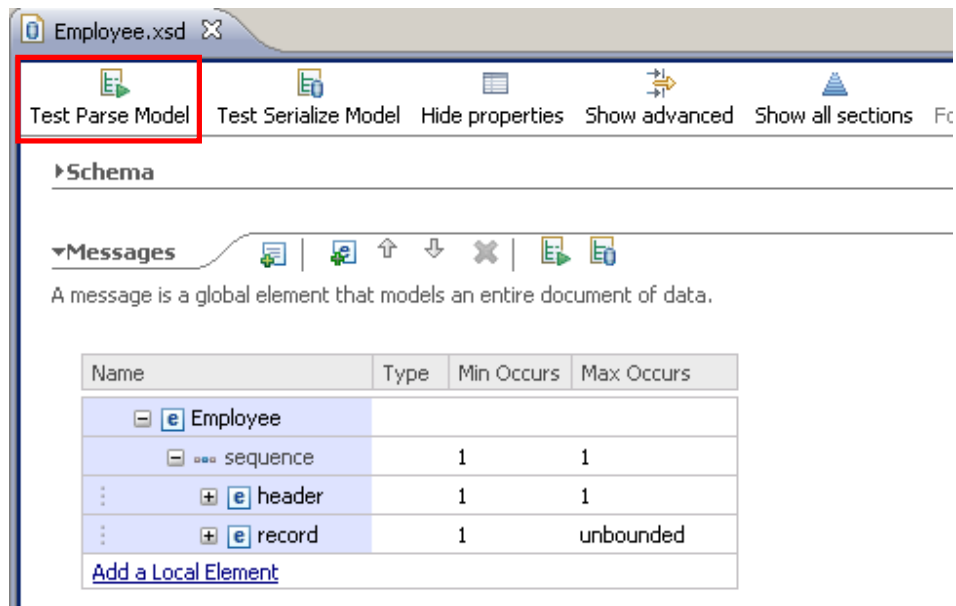
## 16. Save the Employee message model.



## 4. Test the Message Model

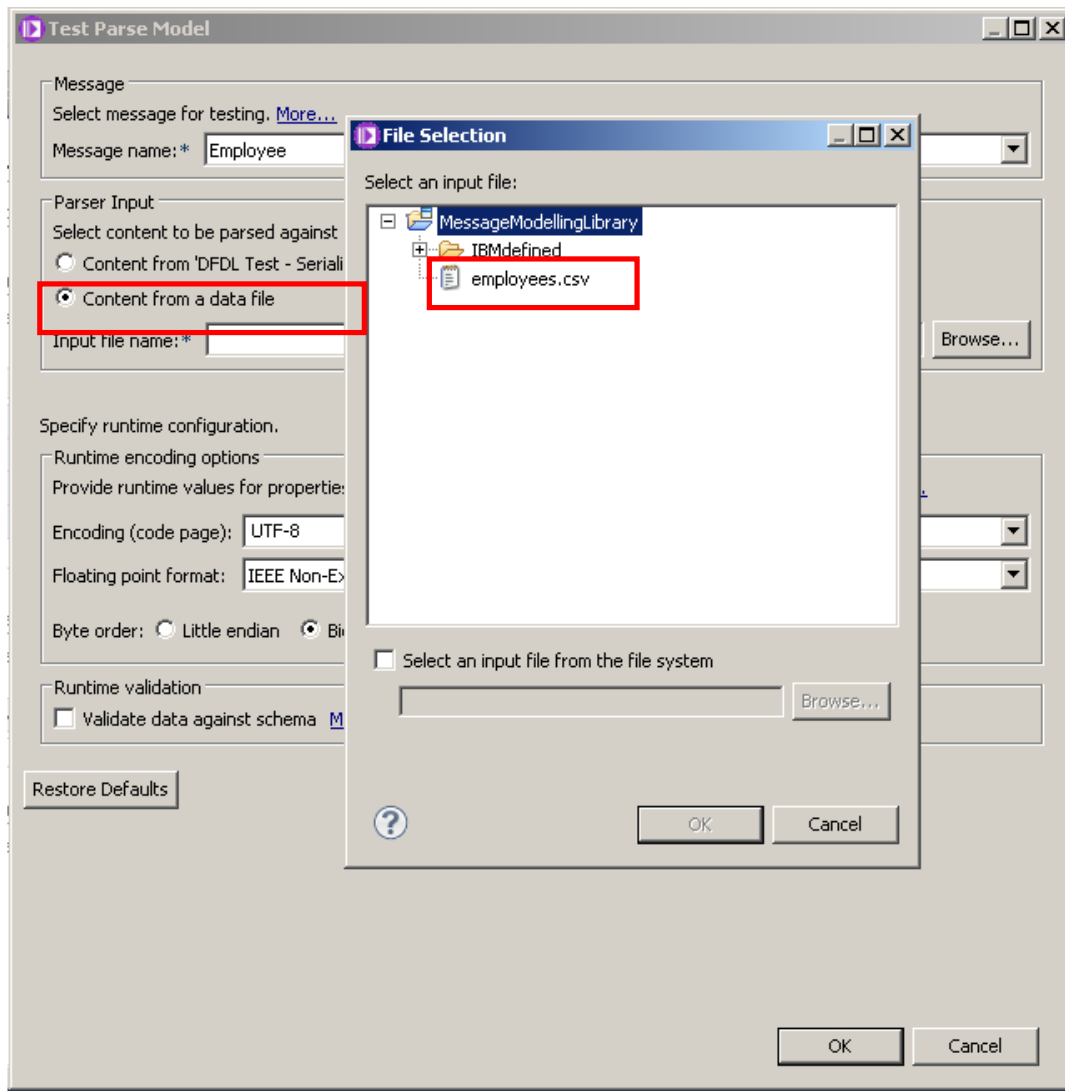
1. Now test the message model to validate that it parses the CSV data file correctly.

Click the "Test Parse Model" icon.



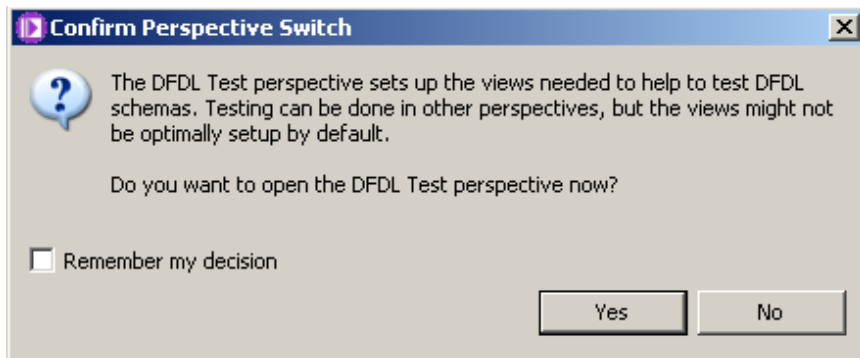
2. In the "Parse Input" section of the new window select the "Content from a data file" option and click the browse button to select the employees.csv file under MessageModellingLibrary.

(Your window may show different items to those shown in the screen capture below).



Then click OK in both windows.

3. Confirm the popup by clicking yes, which will take you to the DFDL Test perspective



#### 4. The DFDL Test perspective has several views:

- DFDL Test - Parse: Allows you to parse a file using the message model in the editor
- DFDL Test - Trace: Has a detailed trace log of the testing activities. Very helpful to find the cause of errors.
- DFDL Test - Logical Instance: Gives you a view of the parsed message tree
- DFDL Test - Serialize: Allows you to generate a file from the message model

When the perspective opens, you see a popup message with the parsing result, in this case successful. Close it by clicking in the "X".

The screenshot shows the DFDL Test perspective in IBM Integration Bus. The main window displays a schema for 'Employee' with the following structure:

Name	Type	Min Occurs	Max Occurs
Employee		1	1
sequence		1	1
header		1	1
sequence		1	1
CompanyName	string	1	1
EmpCount	string	1	1
Address	string	1	1
City	string	1	1
Country	string	1	1
record		1	unbounded

A dialog box titled "Parsing completed successfully." is displayed in the foreground. It includes the following text:

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 Open DFDL Logical Instance View button.
- To view the logical instance that was created by the DFDL parser, click the Open DFDL Logical Instance View button.
- To view the trace captured while running the DFDL parser, click the Open DFDL Trace View toolbar button, or click the Open DFDL Trace View toolbar button.

Below the dialog, the "DFDL Test - Parse" view shows the following information:

Status: Parsing completed: Tue May 22 14:23:43 BST 2012

Input: /MessageModellingLibrary/employees.csv

Parsed Input:

```

Characters
1 Company1,42,123 Street,Winchester,UK
2 "000010" "CHRISTINE" "I" "HAAS" "A00" "3978" "1995-01-01" "PRES" "18" "F" "1963-08-24" 152750.00
3 "000020" "MICHAEL" "L" "THOMPSON" "B01" "3476" "2003-10-10" "MANAGER" "18" "M" "1978-02-02" 94250.00

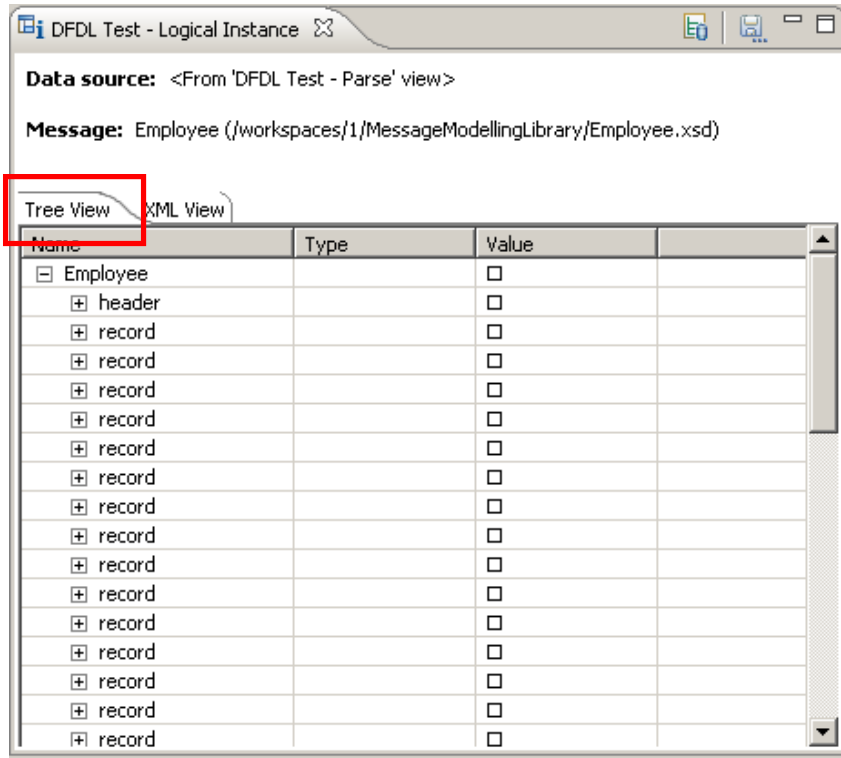
```

Selection in DFDL Editor: Selected: Employee : <Anonymous> (complex) Repeating index: 1 Range in parsed input: 0 - 4171 Selection in Input: Row: 2 Column: 99 Offset: 136 Length: 0

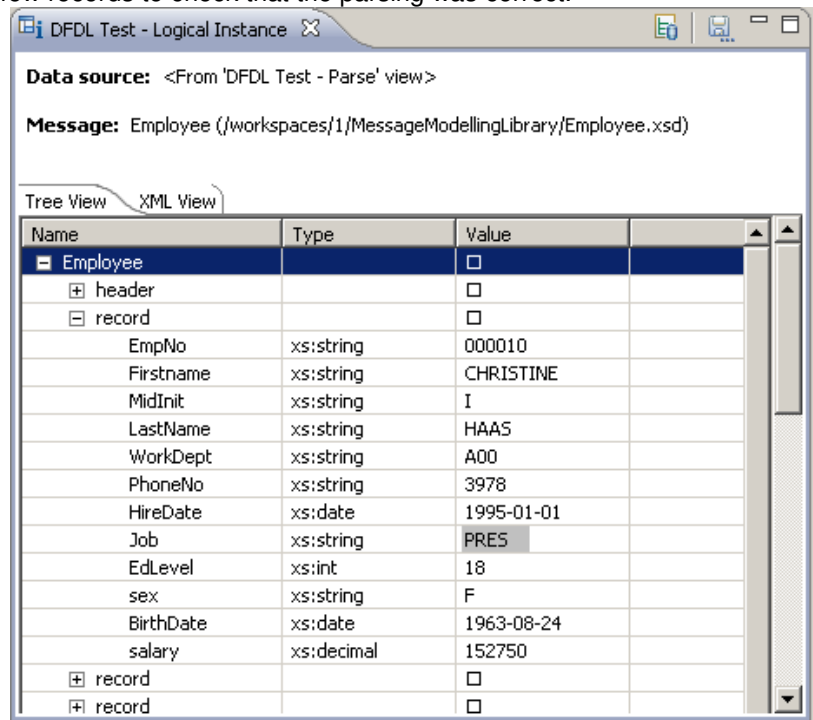
5. Notice that the input text has been highlighted to differentiate the separators (",") that the parser has detected.

```
Parsed Input
Characters
1 Company1,42,123 Street,Winchester,UK
2 "000010", "CHRISTINE", "I", "HAAS", "A00", "3978", "1995-01-01", "PRES", "18", "F", "1963-08-24", 152750.00
3 "000020", "MICHAEL", "L", "THOMPSON", "B01", "3476", "2003-10-10", "MANAGER", "18", "M", "1978-02-02", 94250.00
4 "000030", "SALLY", "A", "KWAN", "C01", "4738", "2005-04-05", "MANAGER", "20", "F", "1971-05-11", 98250.00
5 "000050", "JOHN", "B", "GEYER", "E01", "6789", "1979-08-17", "MANAGER", "16", "M", "1955-09-15", 80175.00
6 "000060", "IRVING", "F", "STERN", "D11", "6423", "2003-09-14", "MANAGER", "16", "M", "1975-07-07", 72250.00
```

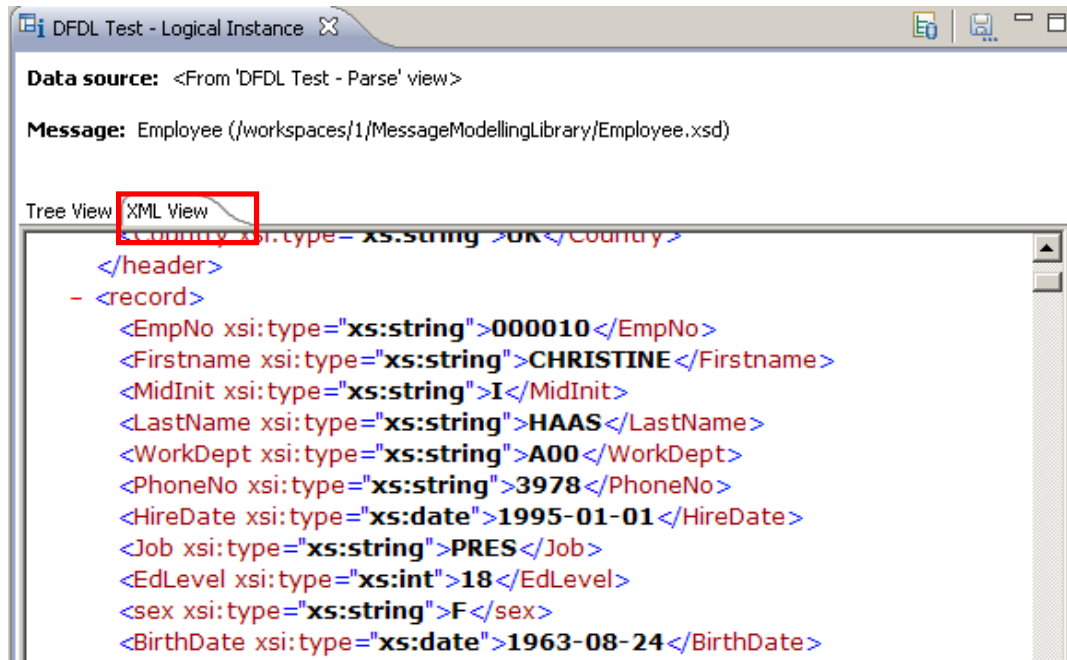
- Look at the Logical instance view in the top right corner. This shows the parsed message tree for the employees.csv file.



Expand a few records to check that the parsing was correct.



7. Click on XML View to show the same parsed records in XML format.



- In the Employee message model, click on the record element. The first record in the input text will be underlined

The screenshot shows the IBM Message Modeling tool interface for the 'Employee.xsd' schema. The 'Messages' section contains a table with the following data:

Name	Type	Min Occurs	Max Occurs
Employee			
sequence		1	1
header		1	1
record		1	unbounded

The 'record' element is highlighted with a red box. Below the table, the 'Parsed Input' section shows the following characters:

```

1 Company1,42,123 Street,Winchester,UK
2 "000010","CHRISTINE","T","HAAS","A00","3978","1995-01-01","PRES","18","F","1963-08-24",152750.00
3 "000020","MICHAEL","L","THOMPSON","B01","3476","2003-10-10","MANAGER","18","M","1978-02-02",94250.00
    
```

The first record (line 2) is underlined. The 'Selection in DFDL Editor' section shows 'Selected: record : <Anonymous> (complex) | Repeating index: 1 | Range in parsed input: 38 - 138'.

To go to another record in the input text, just change the "Repeating Index" number, and it will display the selected record.

The screenshot shows the same IBM Message Modeling tool interface. The 'Repeating Index' in the 'Selection in DFDL Editor' section is now set to 20. The 'Parsed Input' section shows the following characters:

```

20 "000210","WILLIAM","T","JONES","D11","0942","1998-04-11","DESIGNER",17,"M","2003-02-23",68270.00
21 "000220","JENNIFER","R","LUTZ","D11","0672","1998-08-29","DESIGNER",16,"F","1978-03-19",49840.00
22 "000230","JAMES","J","JEFFERSON","D21","2094","1996-11-21","CLERK",14,"M","1980-05-30",42180.00
23 "000240","SALVATORE","M","MARINO","D21","3780","2004-12-05","CLERK",17,"M","2002-03-31",48760.00
24 "000250","DANTEI","S","SMITH","D21","0961","1999-10-30","CLERK",15,"M","1969-11-12",49180.00
    
```

The 20th record (line 20) is underlined. The 'Selection in DFDL Editor' section shows 'Selected: record : <Anonymous> (complex) | Repeating index: 20 | Range in parsed input: 1923 - 2021'.



- You can also click on any element in the message model to see where it's located in the input text.

The screenshot displays the IBM Integration Bus interface. At the top, the 'Employee.xsd' schema is shown in a tree view. The 'Messages' section contains a table with the following data:

Name	Type	Min Occurs	Max Occurs
Employee			
sequence		1	1
header		1	1
record		1	unbounded
sequence		1	1
EmpNo	string	1	1
Firstname	string	1	1
MidInit	string	1	1
WorkDept	string	1	1
PhoneNo	string	1	1
HireDate	date	1	1

Below the schema, the 'DFDL Test - Parse' window is open. The 'Input' section shows the data source as '/MessageModellingLibrary/employees.csv'. The 'Parsed Input' section displays the following CSV data:

```

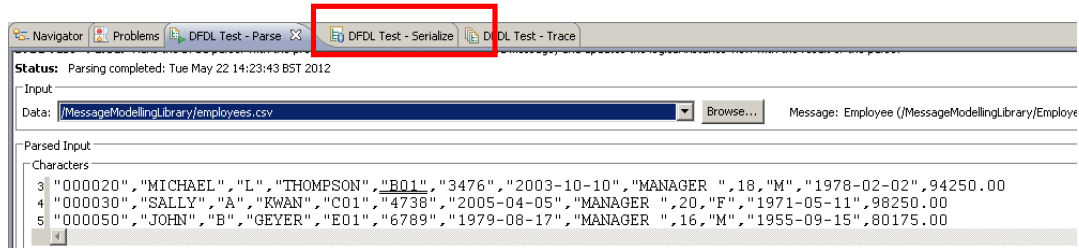
3 "000020","MICHAEL","L","THOMPSON","B01","3476","2003-10-10","MANAGER ",18,"M","1978-02-02
4 "000030","SALLY","A","KWAN","C01","4738","2005-04-05","MANAGER ",20,"F","1971-05-11",98250
5 "000050","JOHN","B","GEYER","E01","6789","1979-08-17","MANAGER ",16,"M","1955-09-15",80175
    
```

The 'WorkDept' element in the schema and the 'B01' value in the CSV data are both highlighted with red boxes. The 'Selection in DFDL Editor' section shows 'Selected: WorkDept : string' and 'Repeating index: 2'. The 'Selection in Input' section shows 'Row: 2 | Column: 99 | Offset: 136 | Length: 0'.

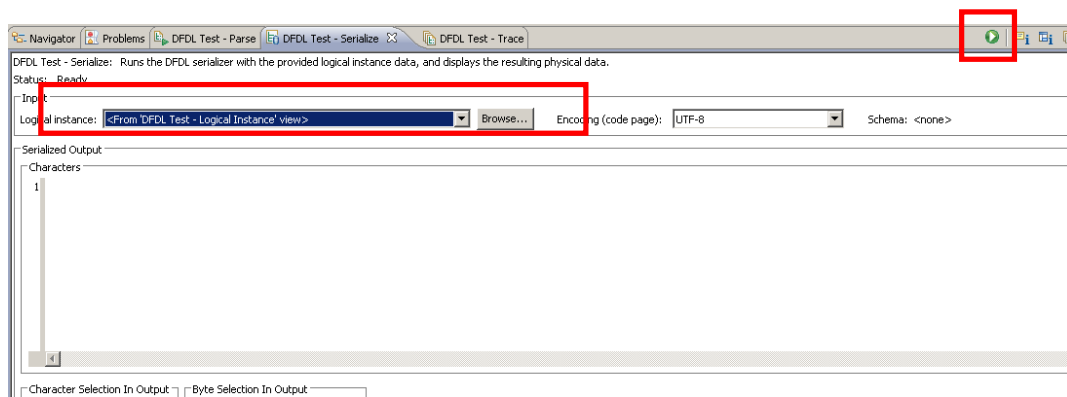
- You could run the parser against other input text, by clicking the Browse button in the input section, selecting the file and clicking the Run Parser button.

The screenshot shows the 'DFDL Test - Parse' window. The 'Input' section has the 'Data' field set to '/MessageModellingLibrary/employees.csv'. The 'Browse...' button next to the data field is highlighted with a red box. In the top right corner of the window, the 'Run Parser' button (represented by a green play icon) is also highlighted with a red box. The 'Parsed Input' section shows the same CSV data as in the previous screenshot.

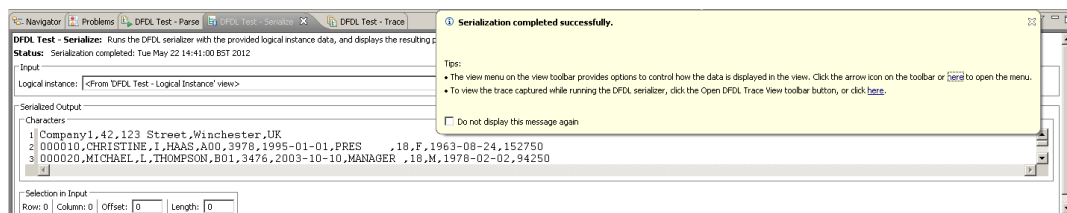
11. Now click on the "DFDL Test – Serialize" tab, next to the DFDL Test – Parse tab.



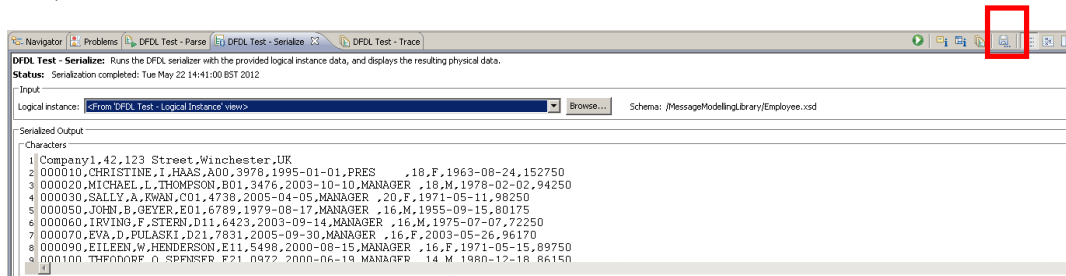
12. In the Input section, click on the combo box and select "<From 'DFDL Test - Logical Instance' view" and click the "Run Serializer" button (the icon on the top right of this view, as highlighted before).



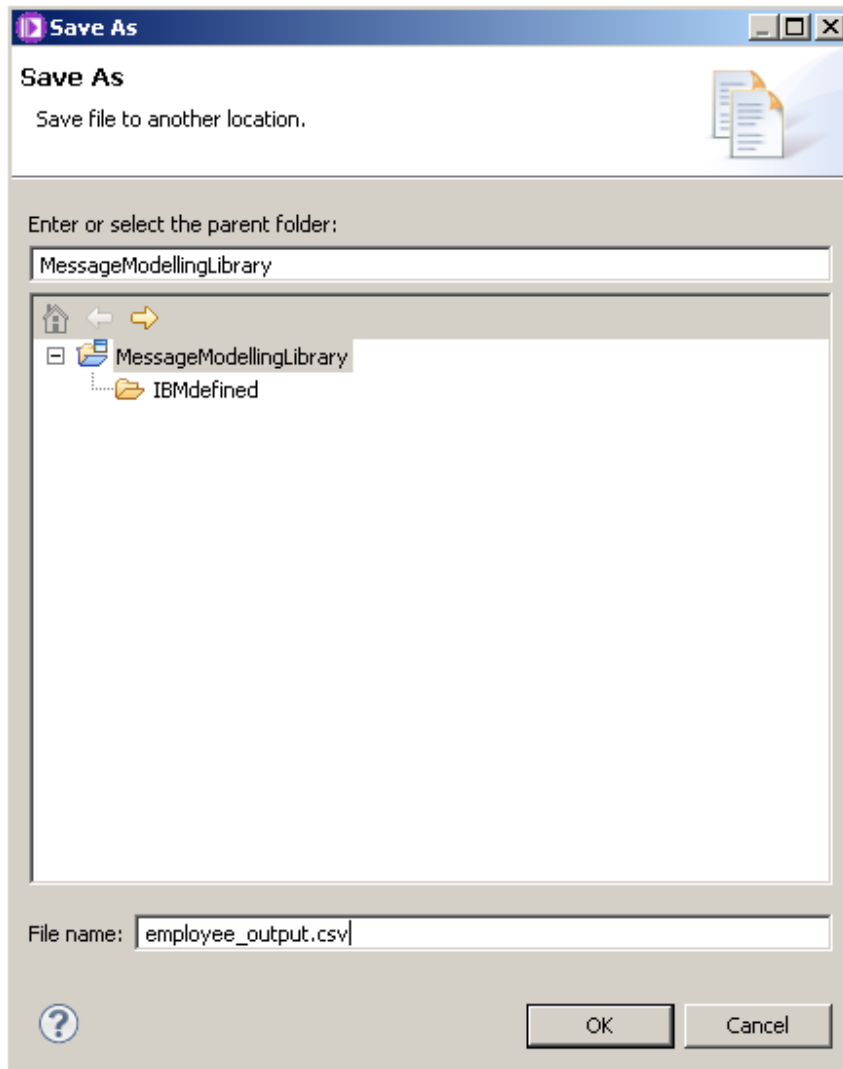
13. The Serializer has just created a text file from the message tree (previously parsed) using the current CSV message model:



14. To save the generated output to a text file, click on the "Save to File" button (the diskette icon).

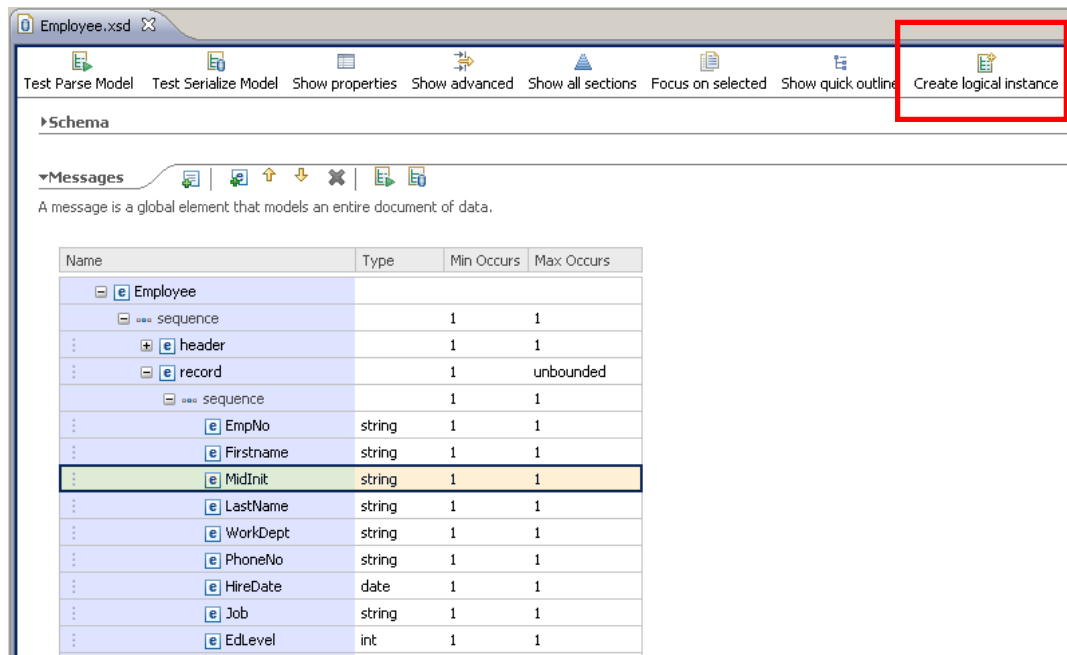


15. In the "Save as" window, select the "MessageModellingLibrary" folder, specify a name for the file (for example: "employee\_output.csv") and click OK.

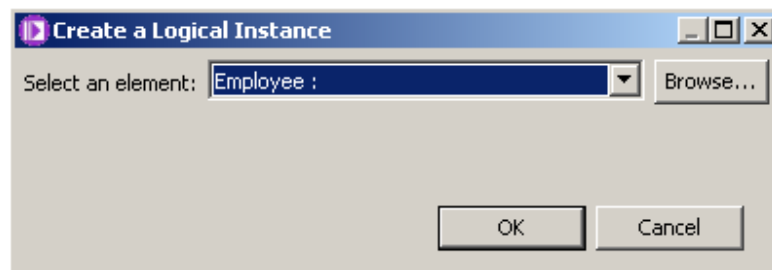


- Now you are going to create a message tree using the "Sample Test Data" values and then serialize it to a file.

Click on the "Create Logical Instance" icon. (You may need to move the slide-bar to the right to expose this icon).



- In the popup, check that "Employee:" is selected and click OK.



- Look at the "DFDL Test - Logical Instance" view (in the right-hand pane), and inspect the message tree.

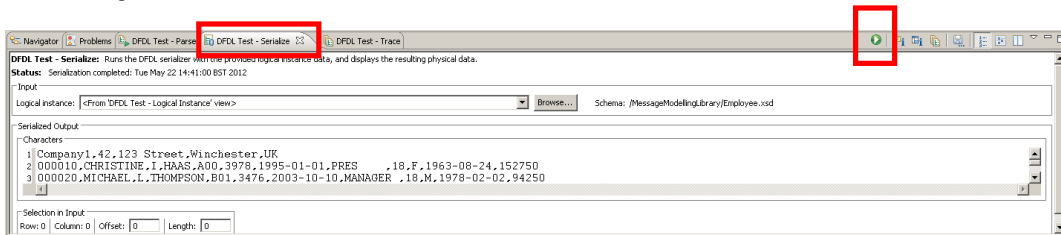
The screenshot shows the IBM Integration Bus interface. The left pane, titled 'Employee.xsd', displays a message tree with the following table:

Name	Type	Min Occurs	Max Occurs
Employee			
sequence		1	1
header		1	1
record		1	unbounded
sequence		1	1
EmpNo	string	1	1
Firstname	string	1	1
MidInit	string	1	1
LastName	string	1	1
WorkDept	string	1	1
PhoneNo	string	1	1
HireDate	date	1	1
Job	string	1	1
EdLevel	int	1	1
sex	string	1	1

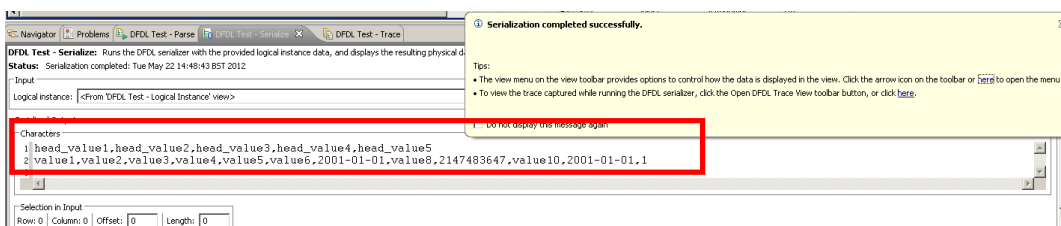
The right pane, titled 'DFDL Test - Logical Instance', shows the 'Tree View' of the logical instance. The data source is '<Generated from DFDL editor>' and the message is 'Employee (/workspaces/1/MessageModellingLibrary/Emplay)'. The tree view shows the following structure:

Name	Type	Value
Employee		
header		
CompanyName	xs:string	head_value1
EmpCount	xs:string	head_value2
Address	xs:string	head_value3
City	xs:string	head_value4
Country	xs:string	head_value5
record		
EmpNo	xs:string	value1
Firstname	xs:string	value2
MidInit	xs:string	value3
LastName	xs:string	value4
WorkDept	xs:string	value5
PhoneNo	xs:string	value6
HireDate	xs:date	2001-01-01
Job	xs:string	value8
EdLevel	xs:int	2147483647
sex	xs:string	value10
BirthDate	xs:date	2001-01-01
salary	xs:decimal	1.0

19. In the "DFDL Test - Serialize" view, make sure the Logical instance is set to "<From 'DFDL Test - Logical Instance' view>". Click on the Run Serializer button.

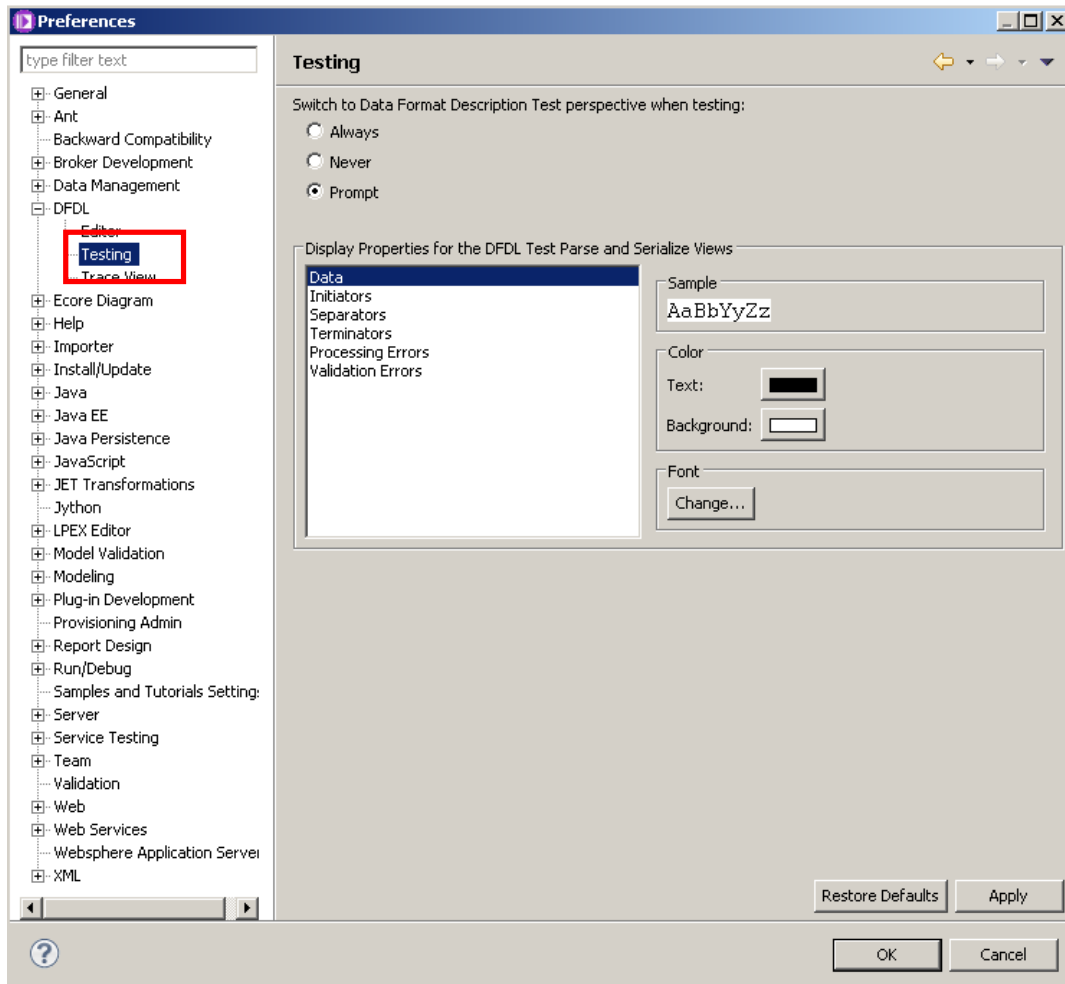


20. The Serializer has created a text file from the message tree generated using the Sample Test Data.



21. Finally, you can configure the DFDL Editor to change the Test Parse configuration properties.

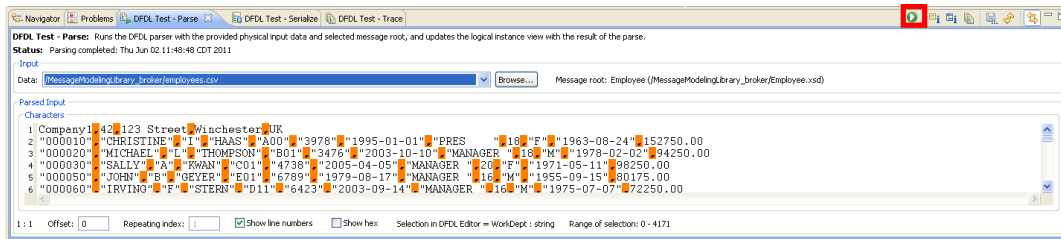
Click Window, Preferences, and from the tree in the left of the window, select DFDL->Testing.



Change the colour for "Separators" to any colour you want, and click OK.



22. Return to the "DFDL Test - Parse" view and click "Run Parser" again:



Notice that now the separators are highlighted with a different colour.

This concludes the CSV Message Modelling lab.