16L91

IBM Integration Bus

Message Modeling with DFDL

Lab 1
Modeling CSV Files

June 2015

Hands-on lab built at product
Version 10.0.0.0

1. INTRODUCTION TO MESSAGE MODELLING 3

1.1 LAB PREPARATION 4

2. CREATING A CSV MESSAGE MODEL 5

3. REFINE THE GENERATED MESSAGE MODEL 23

4. TEST THE MESSAGE MODEL 29

END OF LAB GUIDE 42

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.

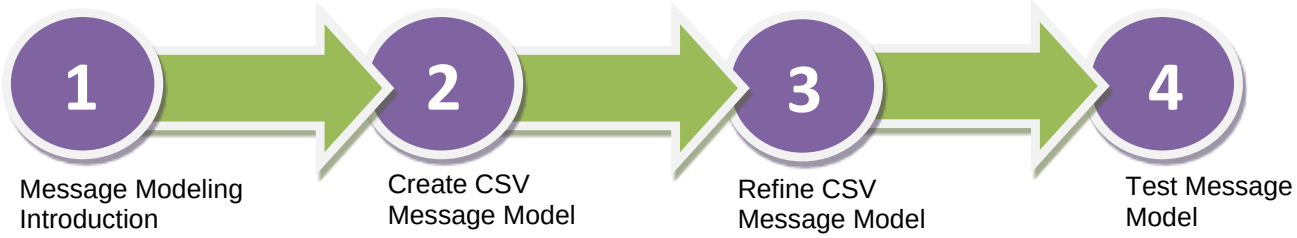
DFDL is not intended to be used to model XML documents. Use normal XML Schema files to model XML.

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:\student10 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.

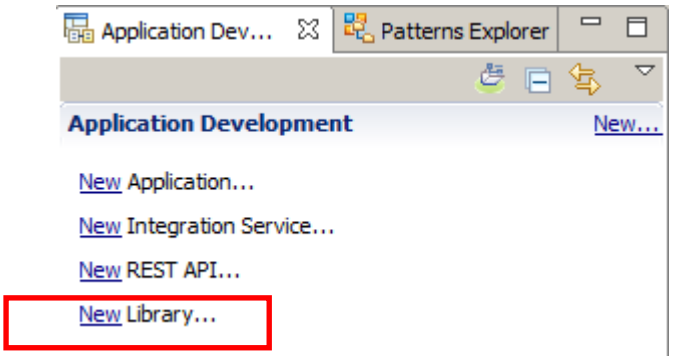
Records are delimited by CRLF characters, and fields within each record are delimited by a comma.

We will use the new Shared Library function that was introduced in IBM Integration Bus Version 10.

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).

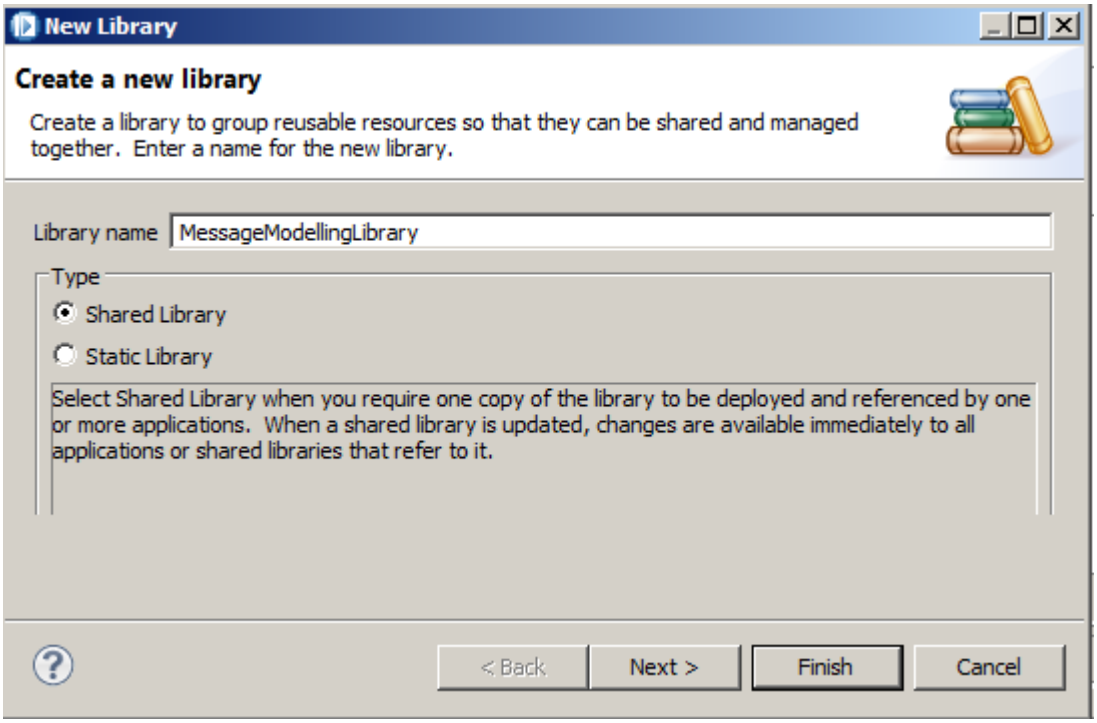
2. Create a new Library by clicking on "New library ….”



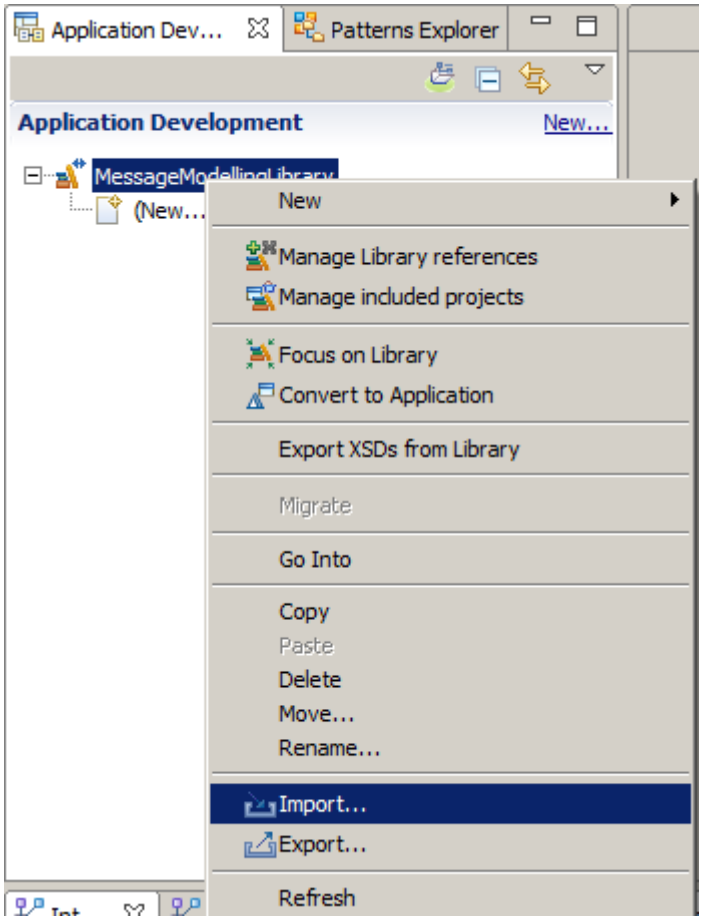
3. You will see radio buttons to choose ‘Shared Library’ or ‘Static Library’. You can choose either, although in this case, we will use a Shared Library.

   Name the library "MessageModellingLibrary".

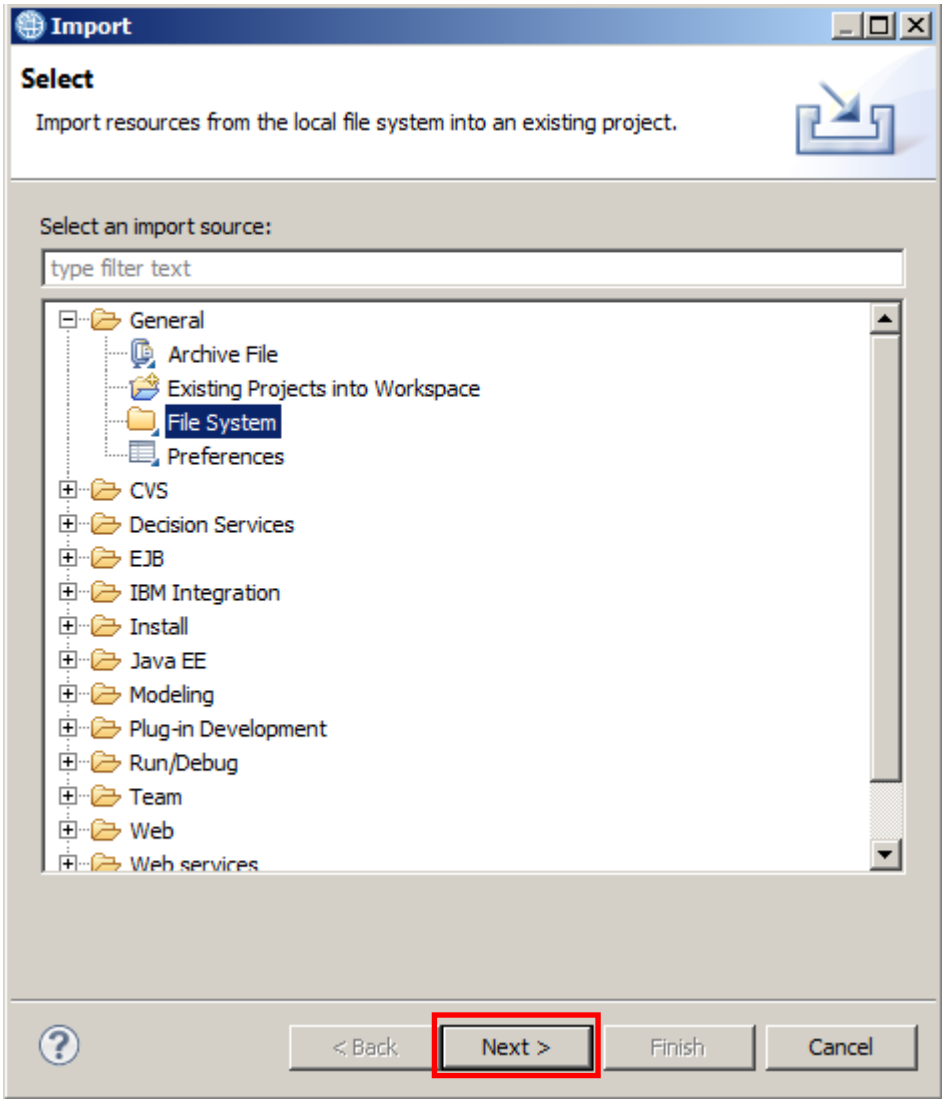
   Click Finish.



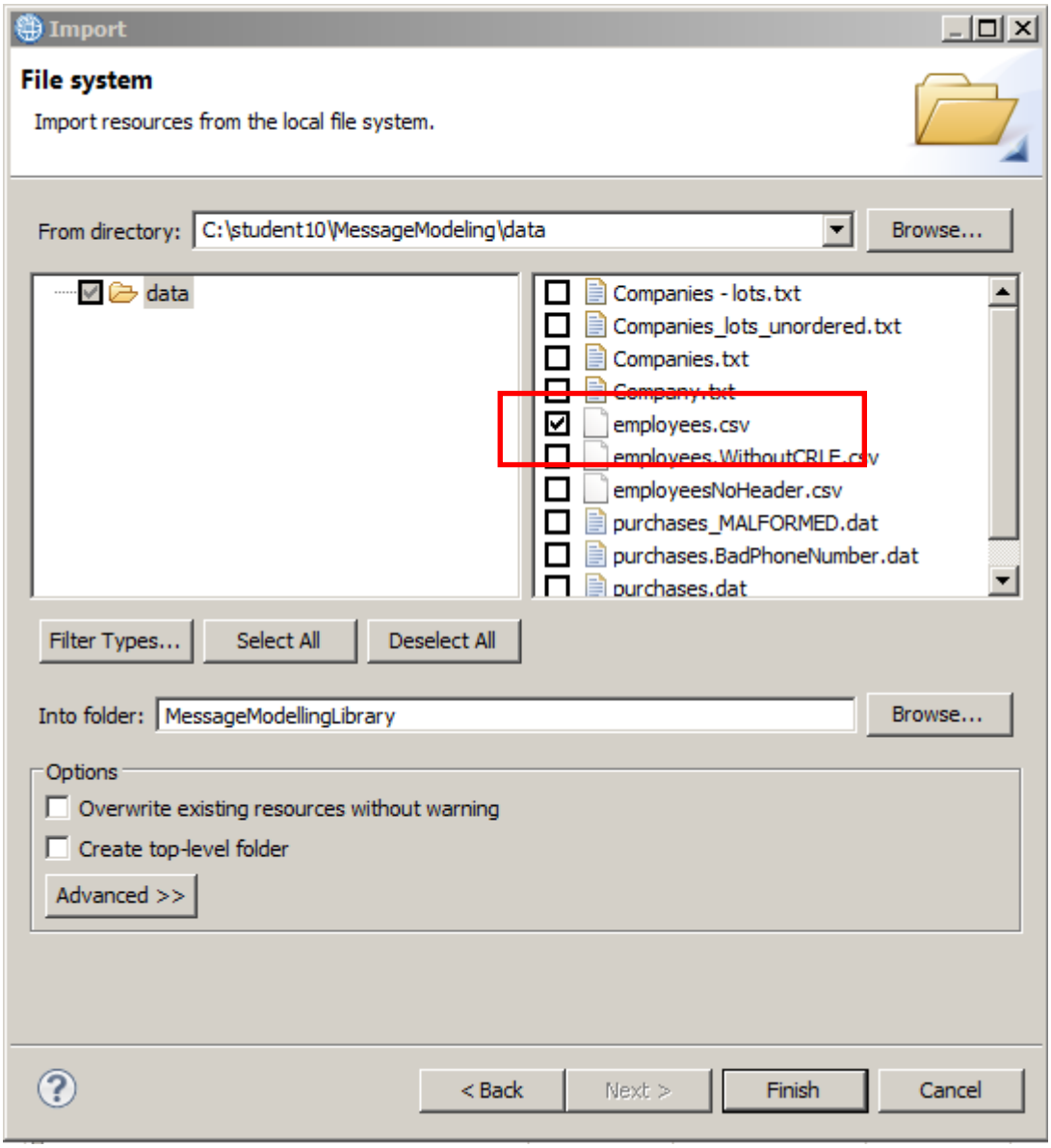
4. Now import the CSV file by right-clicking the generated Integration Bus Project's name (MessageModelingLibrary) and selecting Import.



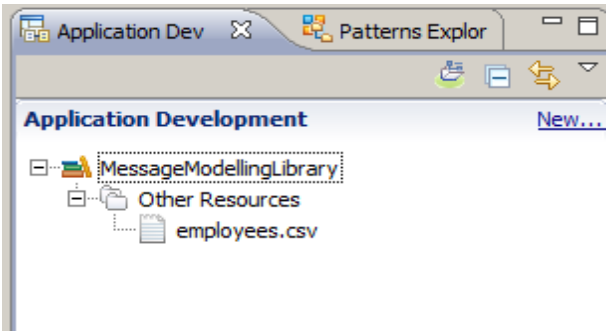
5. In the Import window, select General -> File System and click Next.



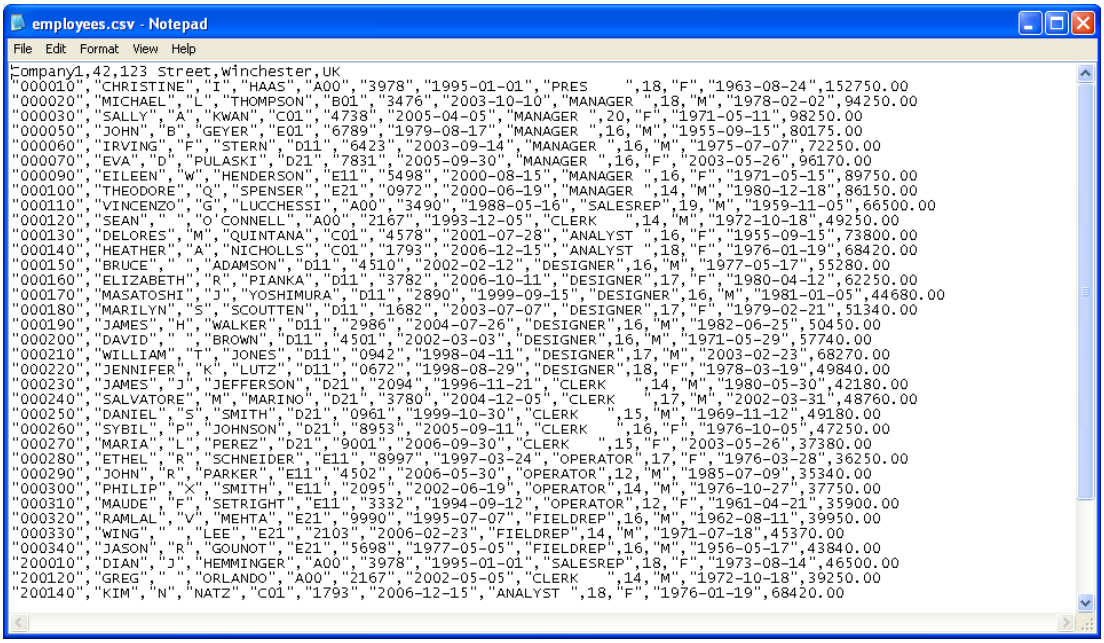
6. Click Browse and select “c:\student10\MessageModelling\data”. In the right side of the window, select “employees.csv” and click Finish.



7. Your project should now look like this:

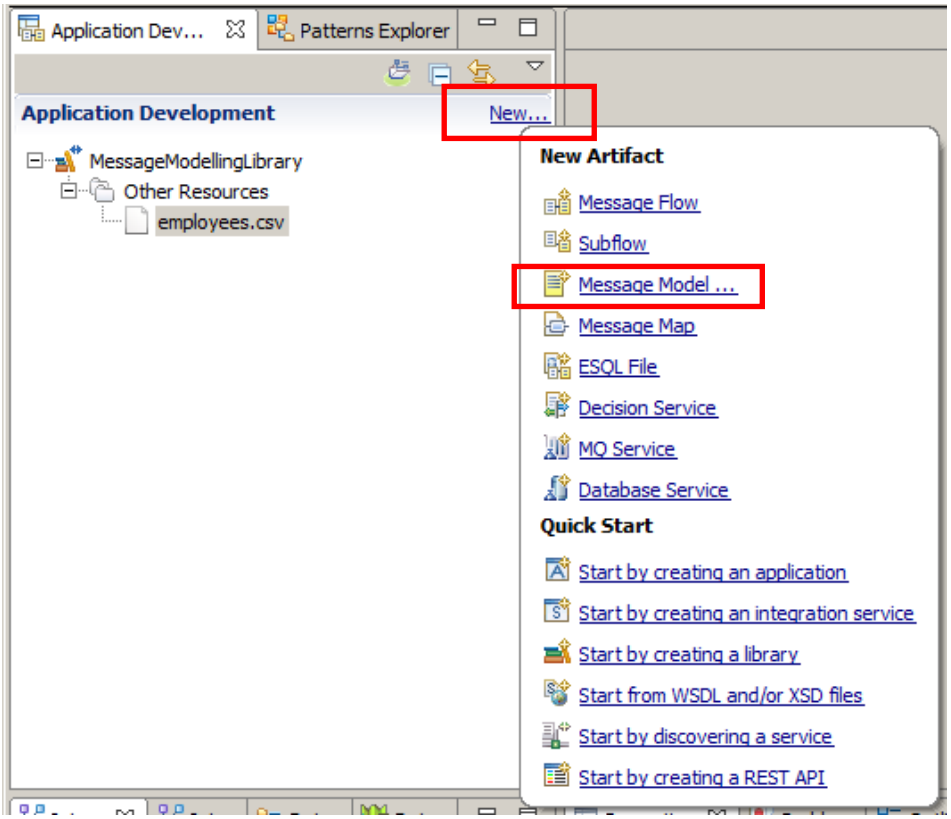


8. Double-click on "employees.csv" to open it, and take a quick look at it. If prompted, select to open it with Microsoft Notepad.



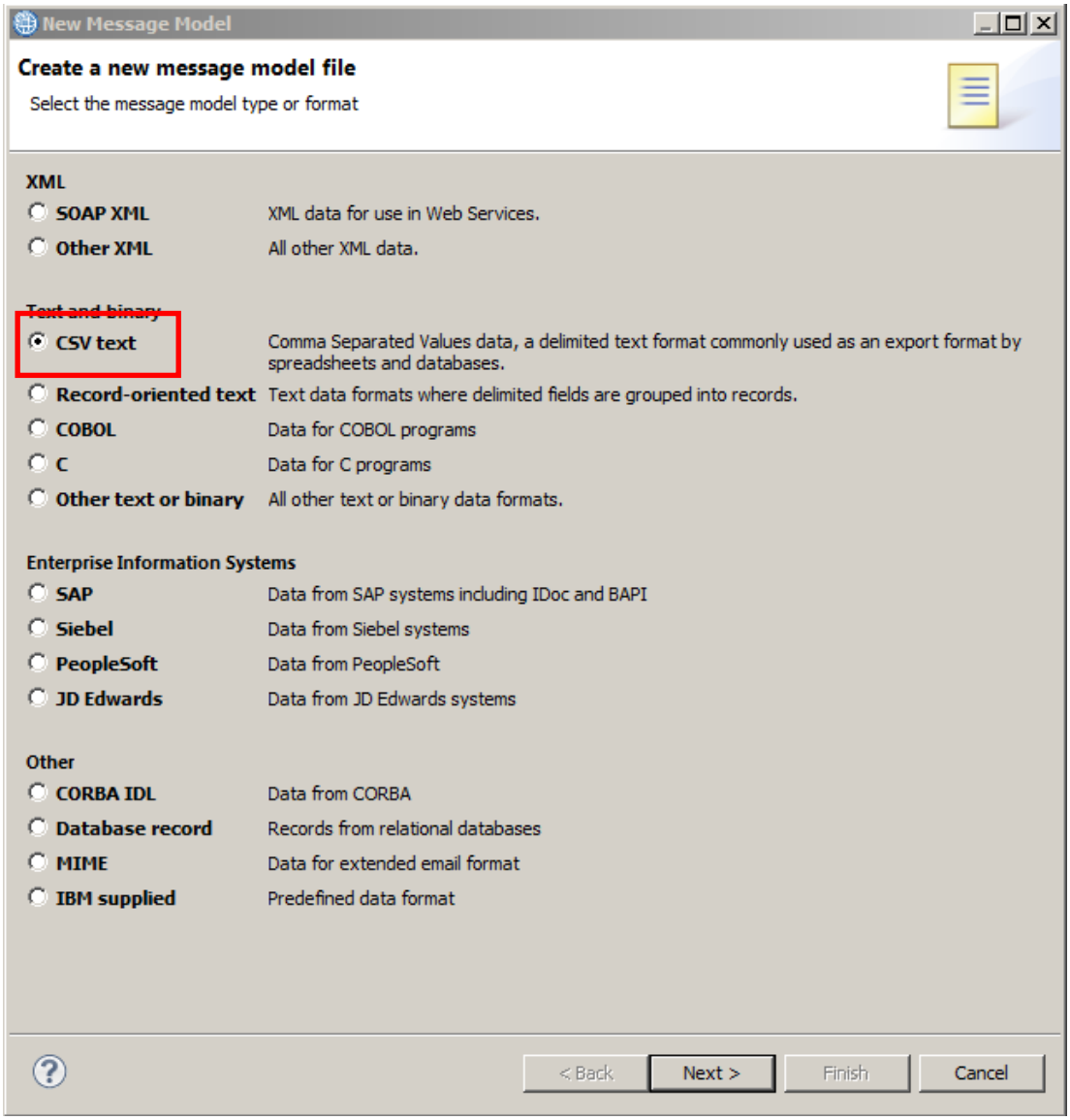
It has a header record, and 42 detail records. Each detail record has 12 fields.

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

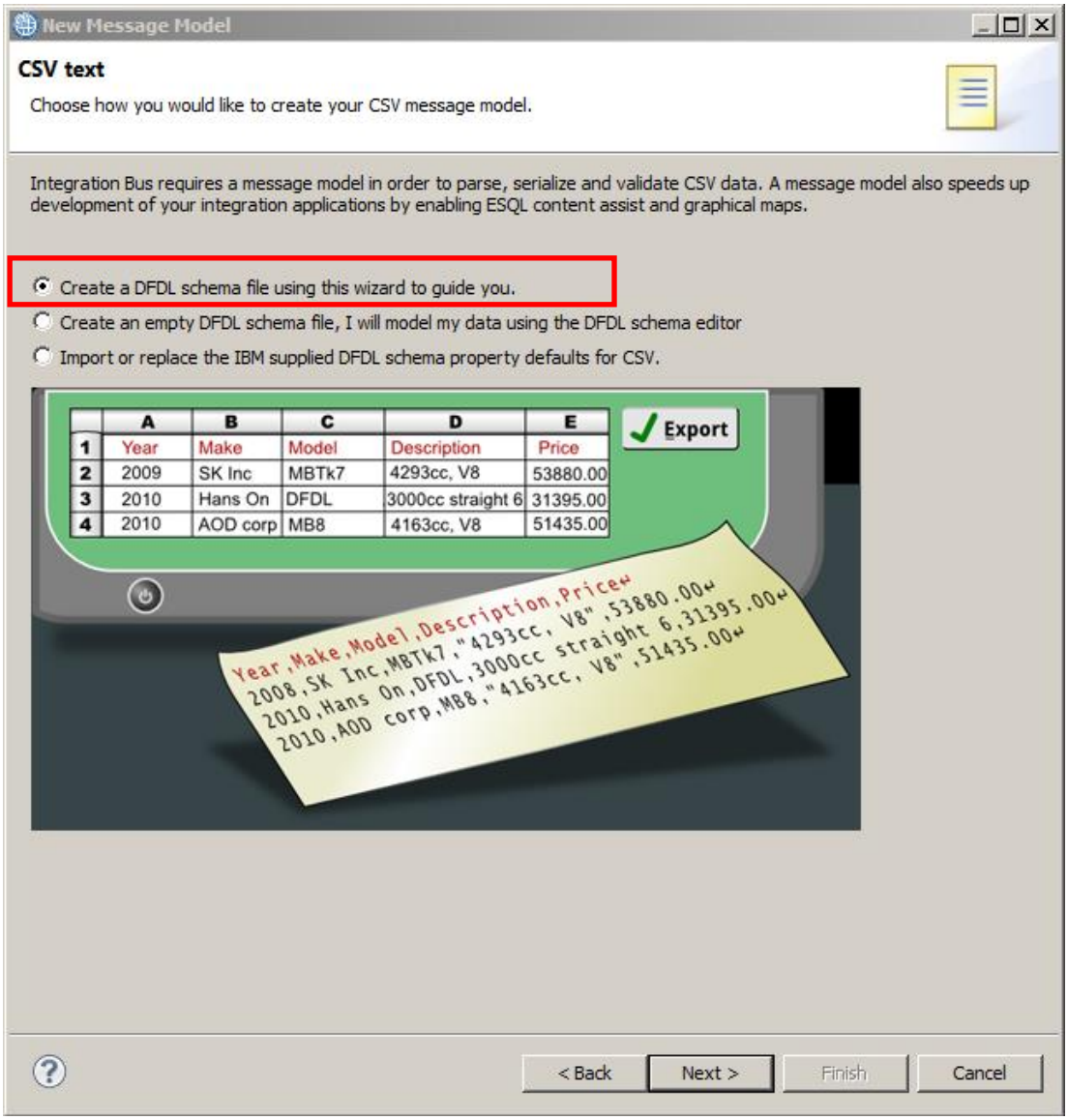


10. This will open up the Message Model creation wizard.

    Select "CSV text", and click Next.



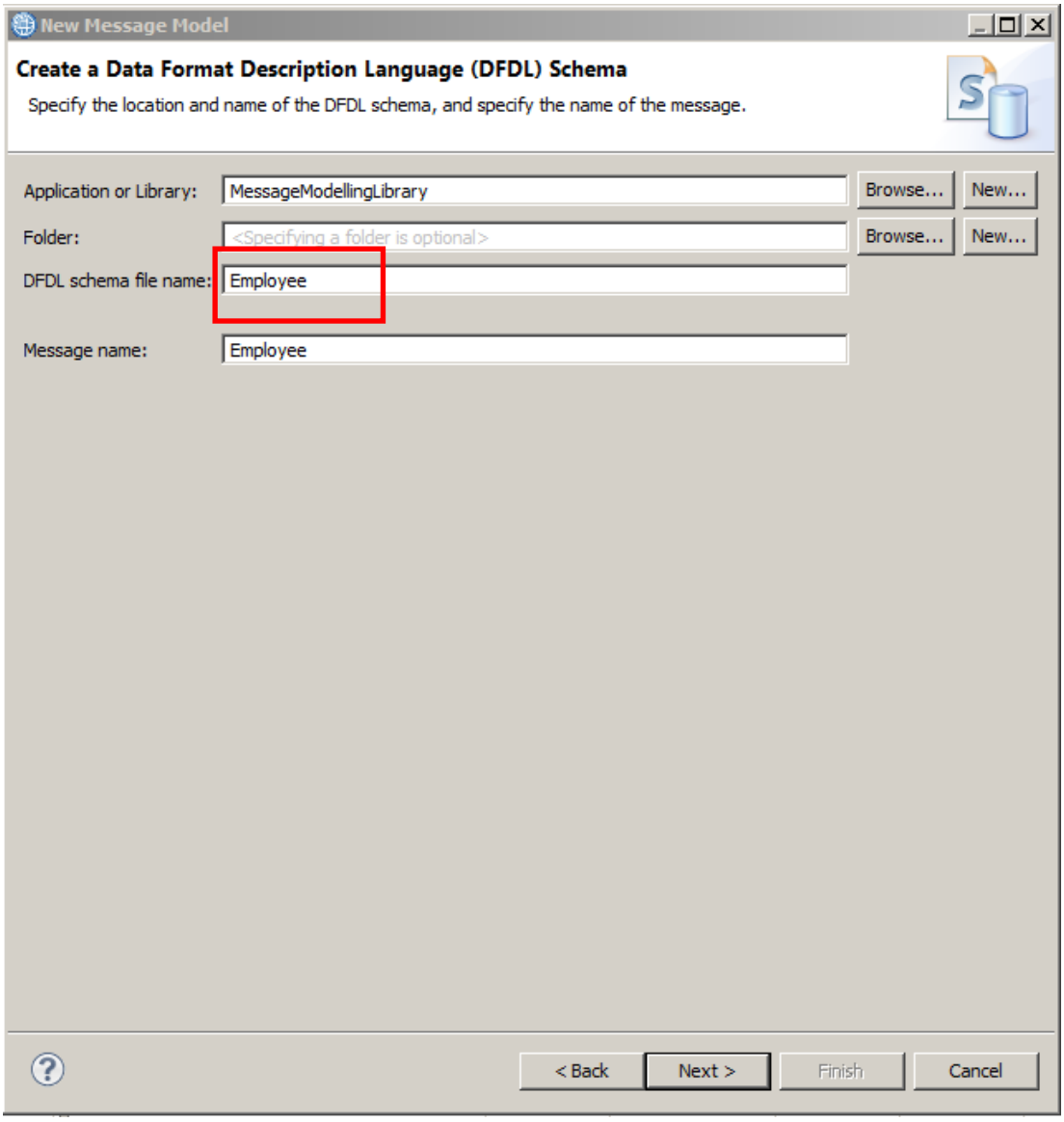
11. Then you can choose to use the CSV Message Model Creation wizard or use the DFDL editor. Select "Create a DFDL Schema file using the wizard to guide you" and click Next.



	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

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

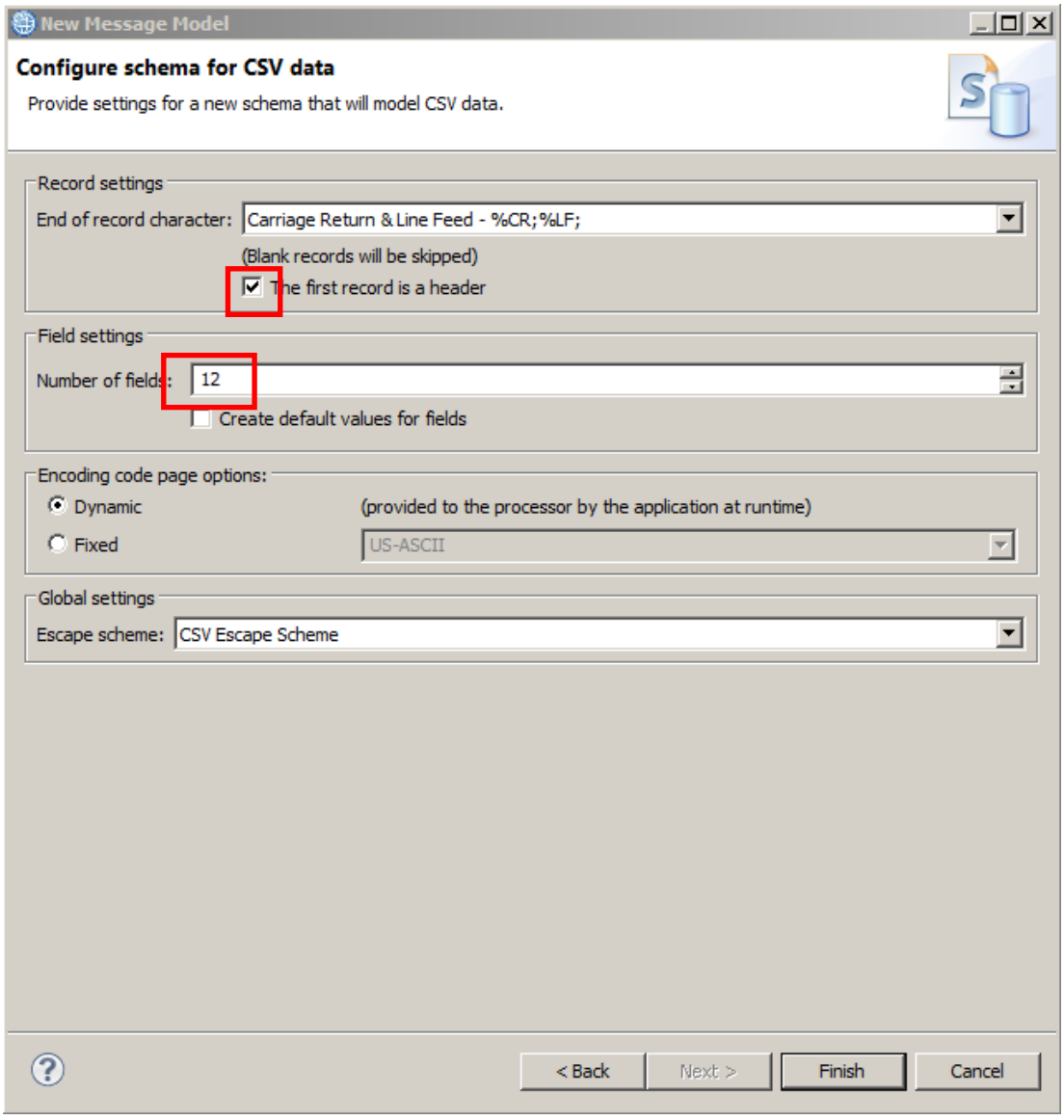
    Enter "Employee" as the name for the DFDL Schema file and click Next.



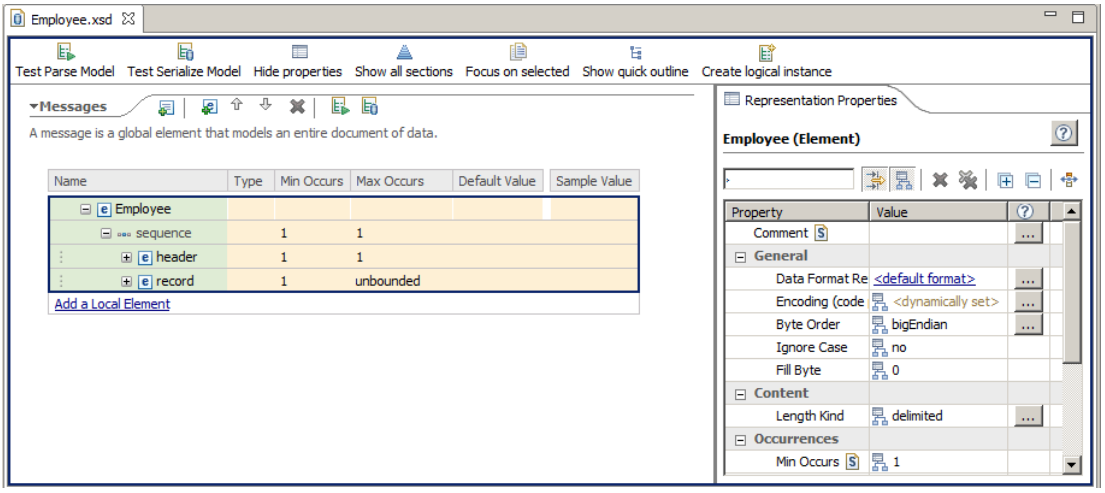
13. In the next screen the wizard lets us choose some settings about the CSV format, like the End of Record character, Blank Records treatment, Number of fields, Encoding and Escape Scheme.

    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.

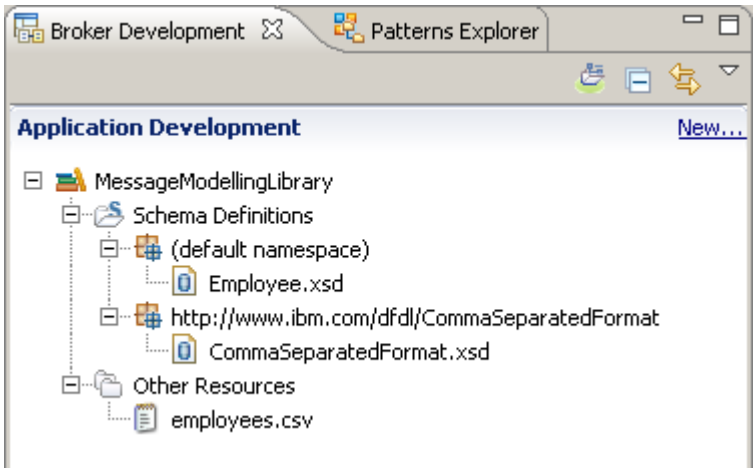


14. Now the DFDL editor should show the CSV message model.



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.

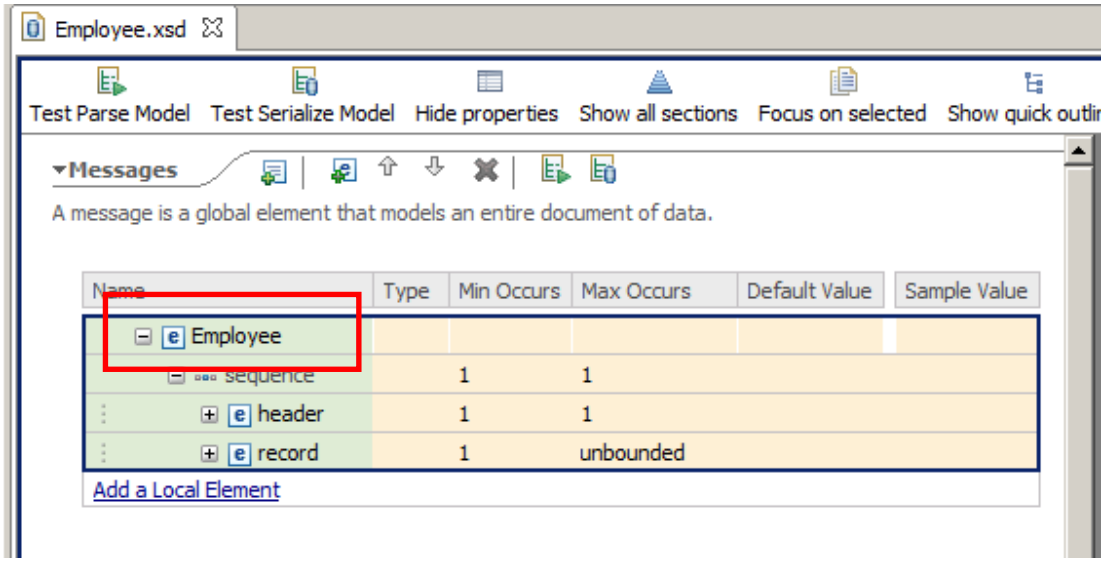


You'll also notice that there is another XSD file under Schema Definitions called "CommaSeparatedformat.xsd". This "Helper schema file" was automatically added by the CSV wizard and contains CSV-specific defaults for all the DFDL properties. This is required because DFDL doesn't have built-in defaults, 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.

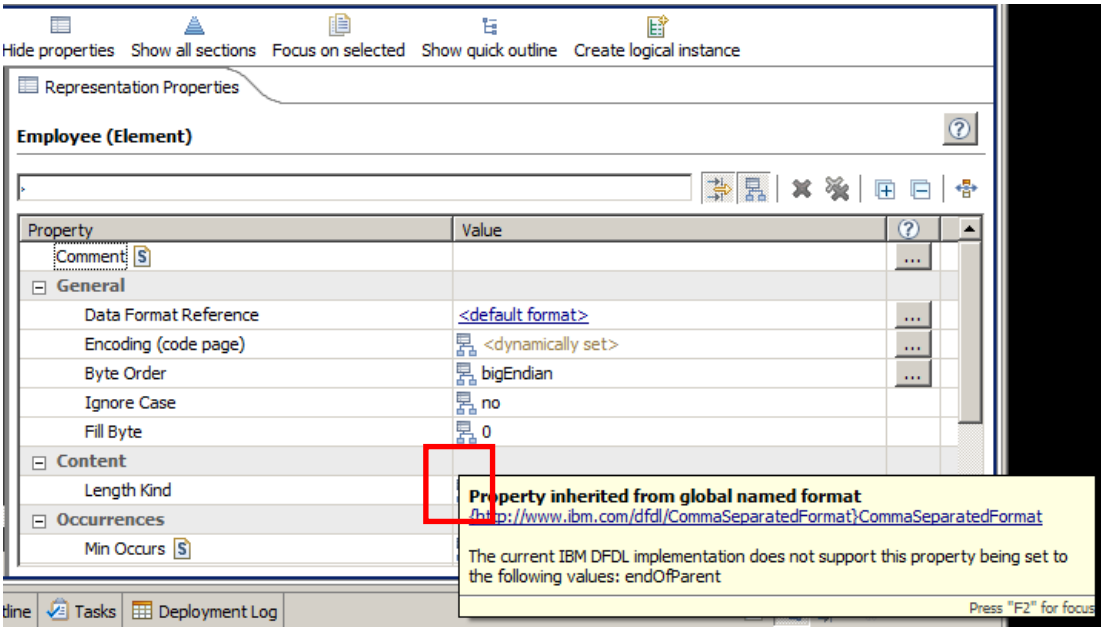
    In the Message Roots view (the left-hand side of the Message Model editor) click on the Employee element.



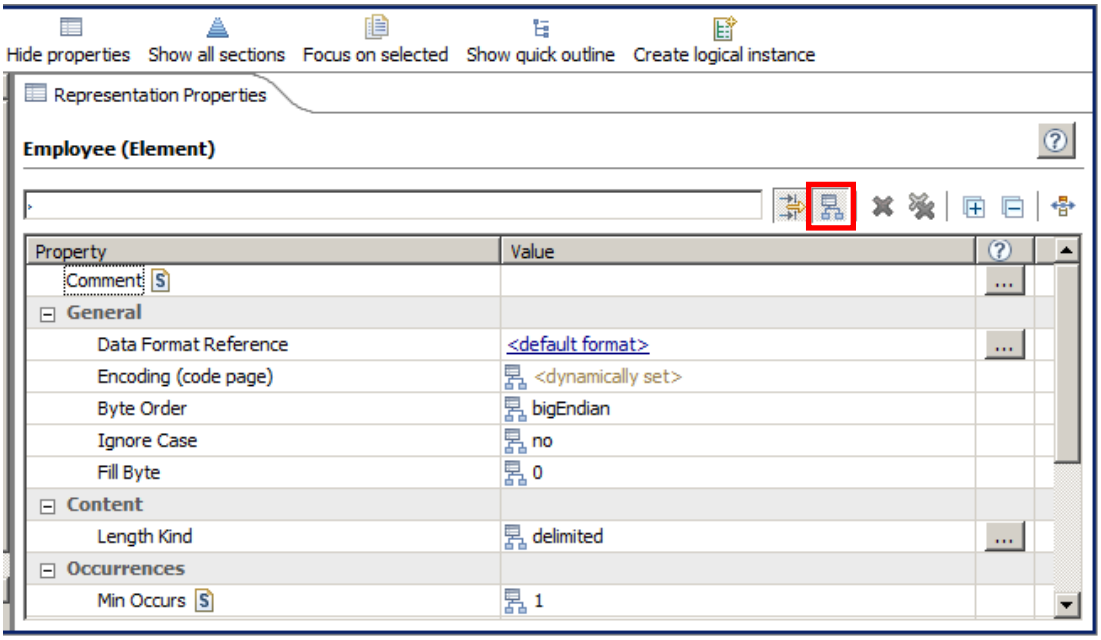
16. Take a look at the Representation Properties view on the right hand side. Notice the "inheritance" icon to the left of the "Length Kind" property.

    Hover over the Inheritance icon to discover its origin.

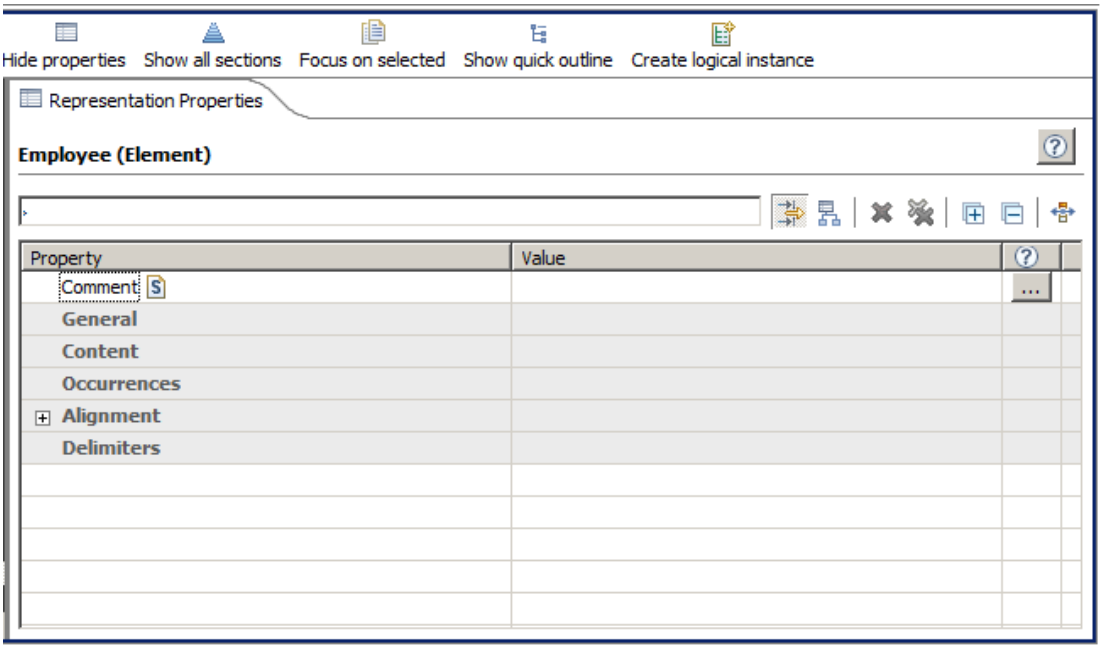
    In this case, the "Length Kind" property was inherited from the "CommaSeparatedFormat" Helper schema that the wizard automatically imported.



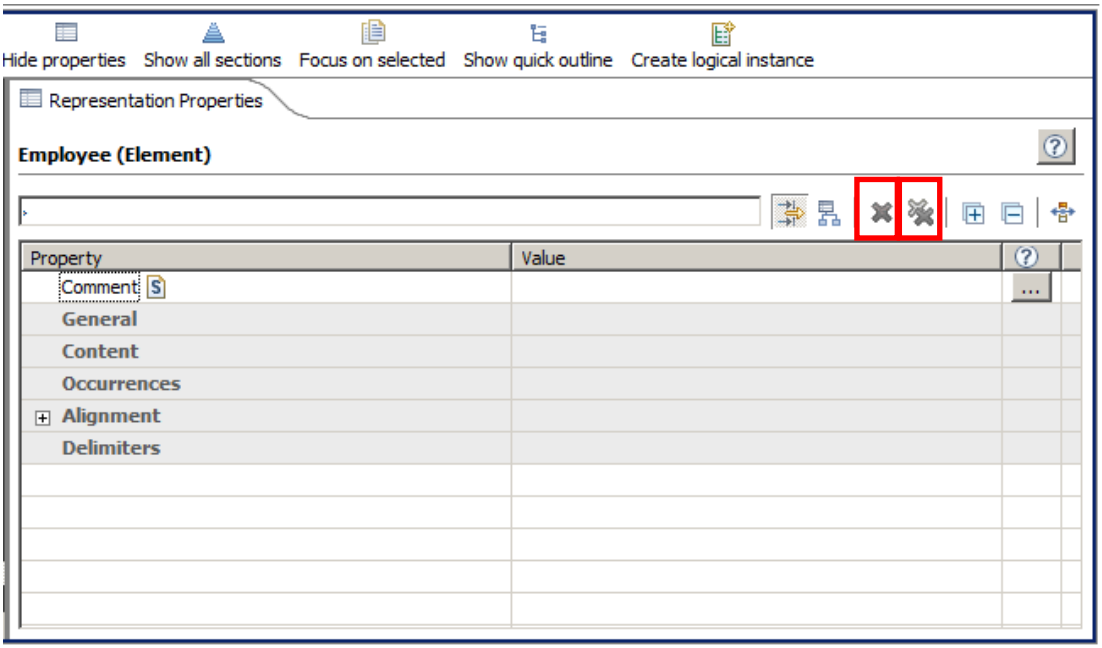
17. Click on the "Show Local Properties" icon.



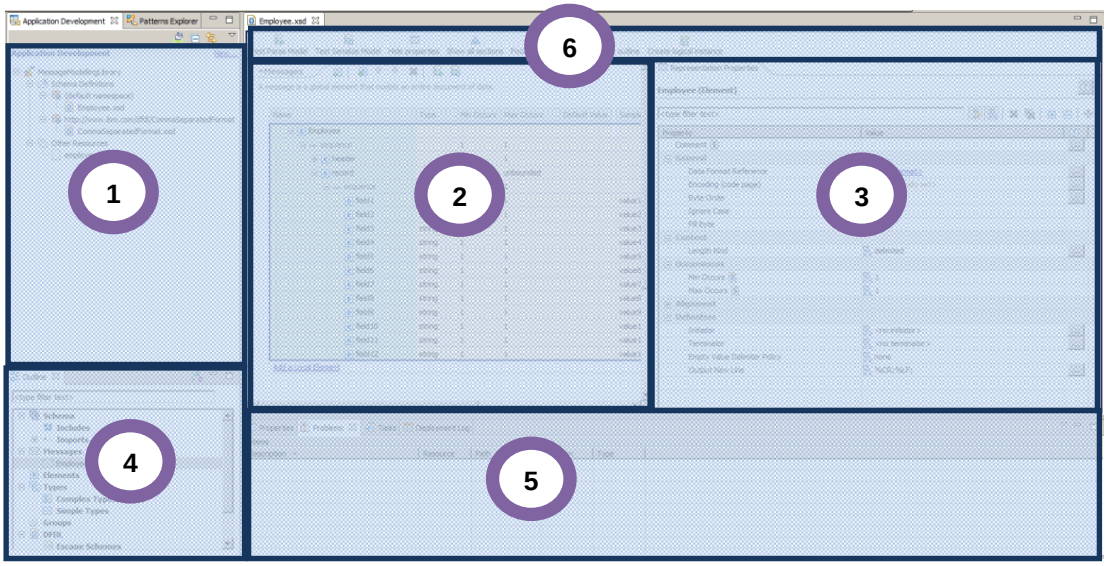
18. Notice that now all the inherited properties are hidden. Only the locally defined properties are shown. (In this case there are none).



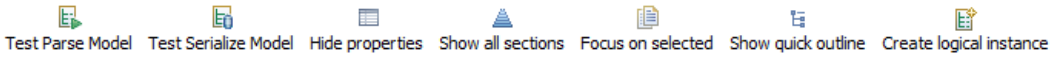
19. The other two icons to the right of the "Show Local Properties"/ "Show Inherited Properties" icon, allow you to clear a selected local property or all of them and replace them with the inherited value.



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

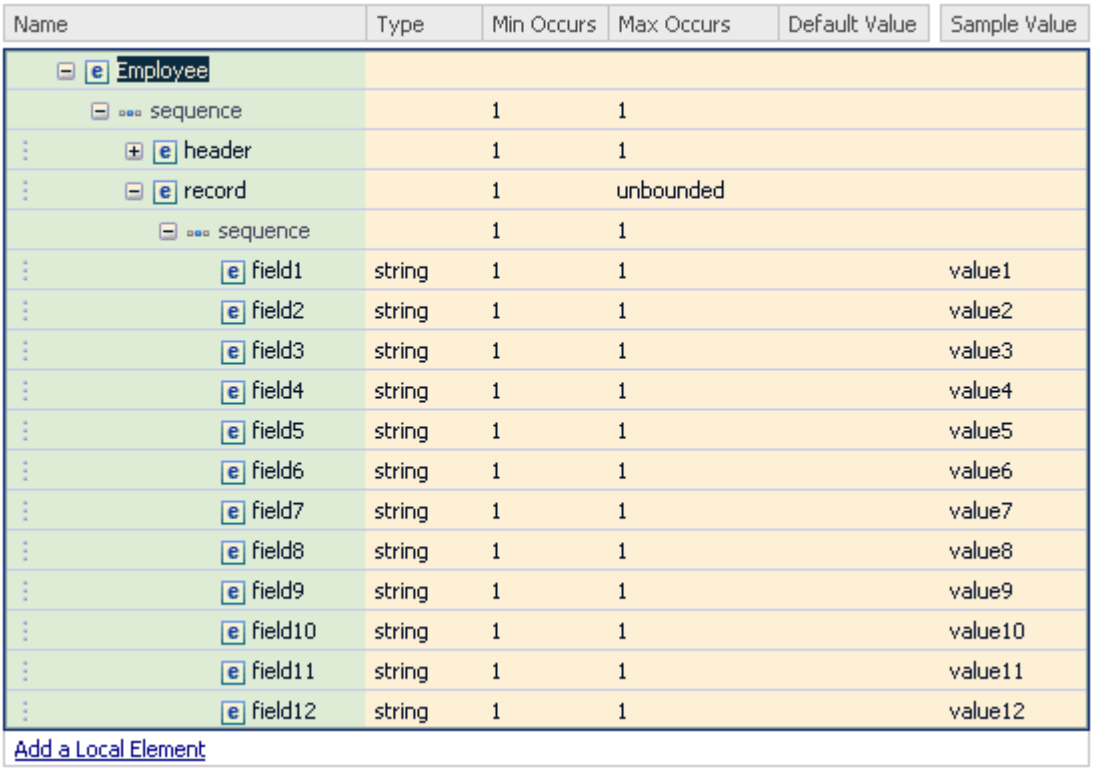


1. **Navigator view:** The Navigator view shows a hierarchical view of all the resources that are currently in your workspace. By expanding the folder for a project, you can see the resources, including any DFDL schema files, that this message set project contains. Selecting a DFDL schema file in this view opens it for editing in the Editor.
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 warnings 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 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
Employee					
sequence		1	1		
header		1	1		
record		1	unbounded		
sequence		1	1		
field1	string	1	1		value1
field2	string	1	1		value2
field3	string	1	1		value3
field4	string	1	1		value4
field5	string	1	1		value5
field6	string	1	1		value6
field7	string	1	1		value7
field8	string	1	1		value8
field9	string	1	1		value9
field10	string	1	1		value10
field11	string	1	1		value11
field12	string	1	1		value12

Add a Local Element

Take a look at the columns:

1. Name: name of the data element
2. Type: data type (string, int, boolean, decimal, date, etc)
3. Min Occurs: minimum amount of occurrences expected (0 or greater)
4. Max Occurs: maximum amount of occurrences expected (from 1 to unbounded)
5. Default value: you can specify a default value for each field
6. 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:

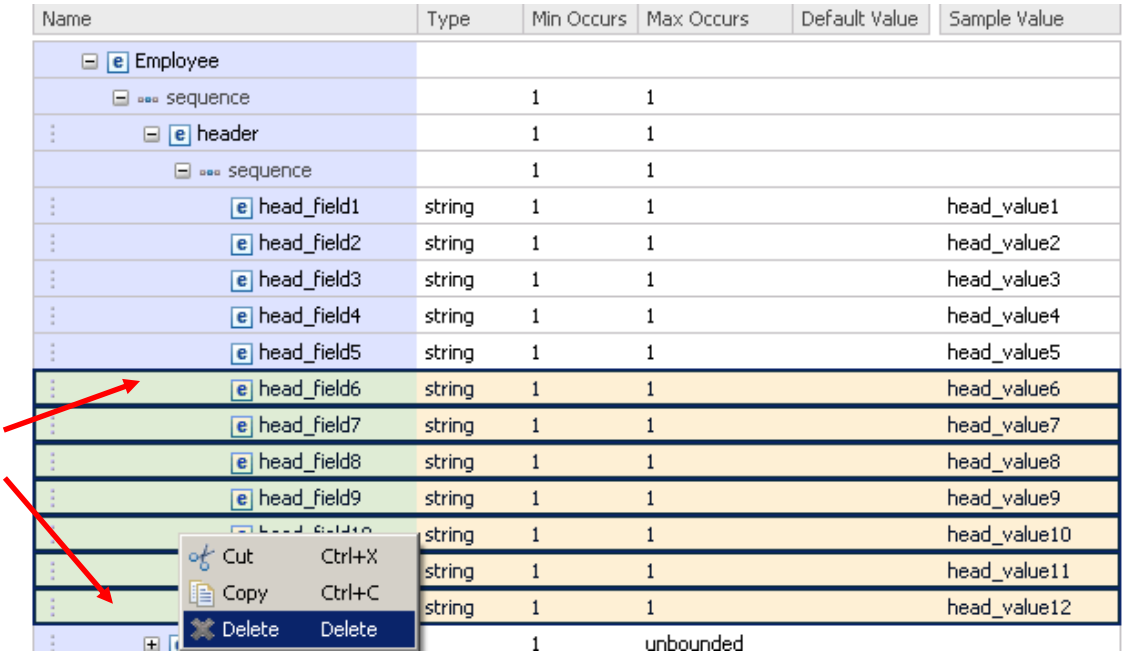
- Adjust the number of elements in the header.
- Change the names of the field elements to reflect the field usage, and adjust the data types of some the elements.

Note that the wizard treats the CRLF delimiter as a Terminator on the header and record elements, rather than a Separator on the Employee element's sequence. This is more flexible, as it allows the case where some instances of the data have no final trailing CR/LF, but other instances do (as we assume in this case), by using 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.

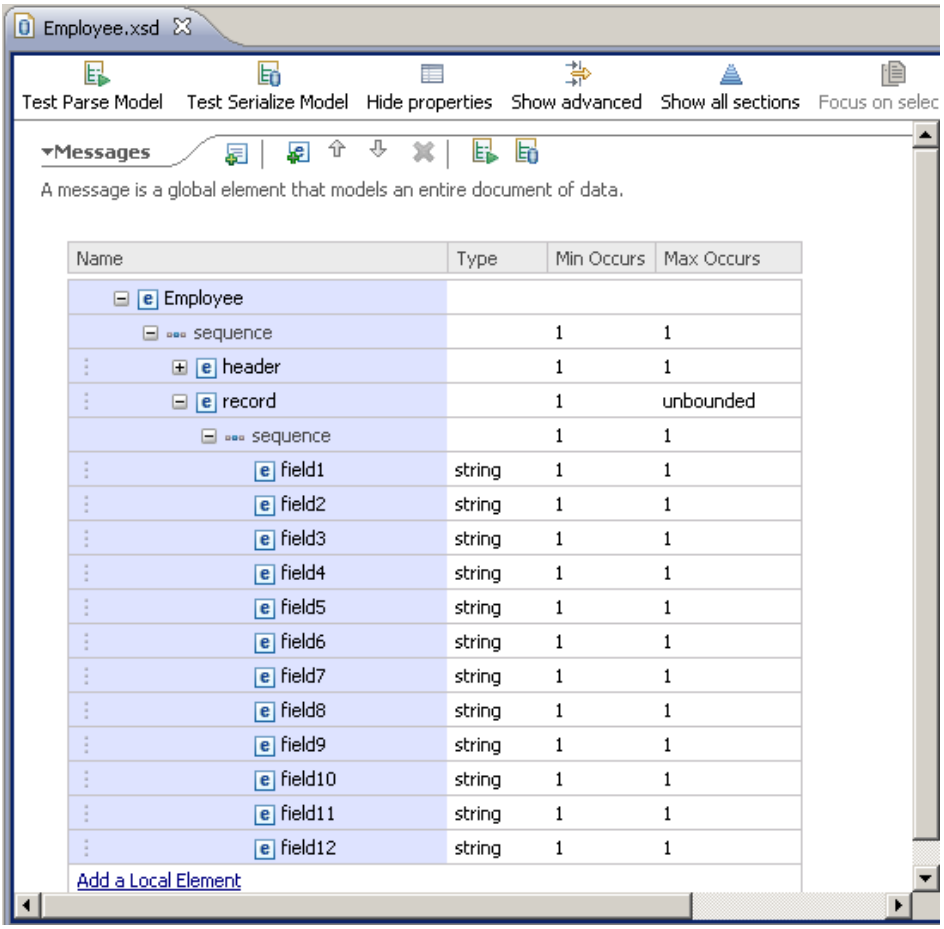
   Select the elements by placing the mouse on the left side of the element, not the element name.



2. 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
CompanyName	string	1	1
EmpCount	string	1	1
Address	string	1	1
City	string	1	1
Country	string	1	1
record		1	unbounded
Add a Local Element			

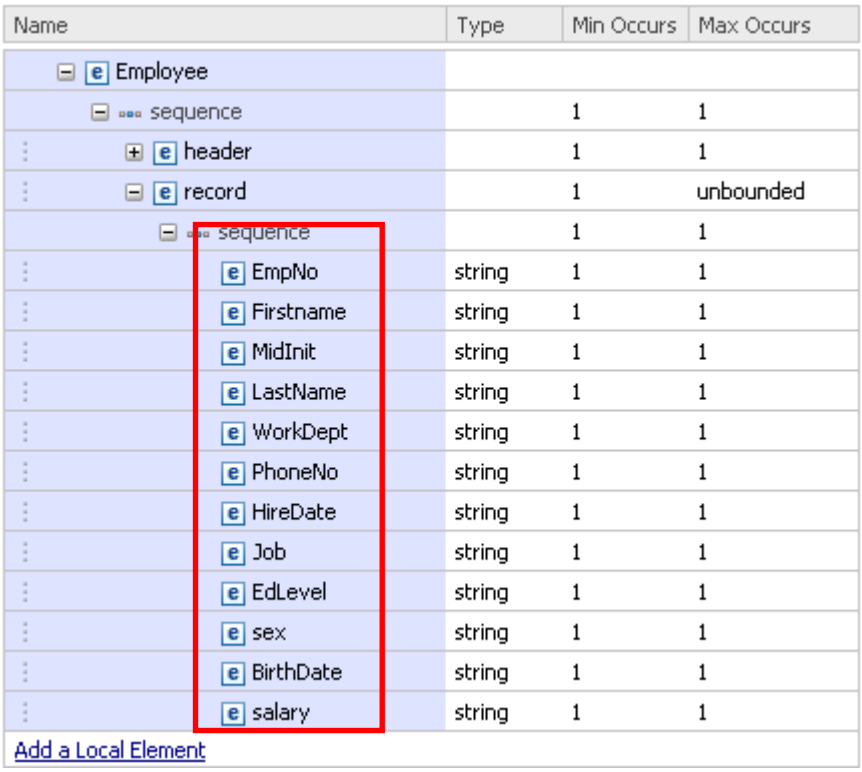
3. Expand the "record" element to show the 12 fields created by the wizard. Change their names, to reflect the CSV field names.



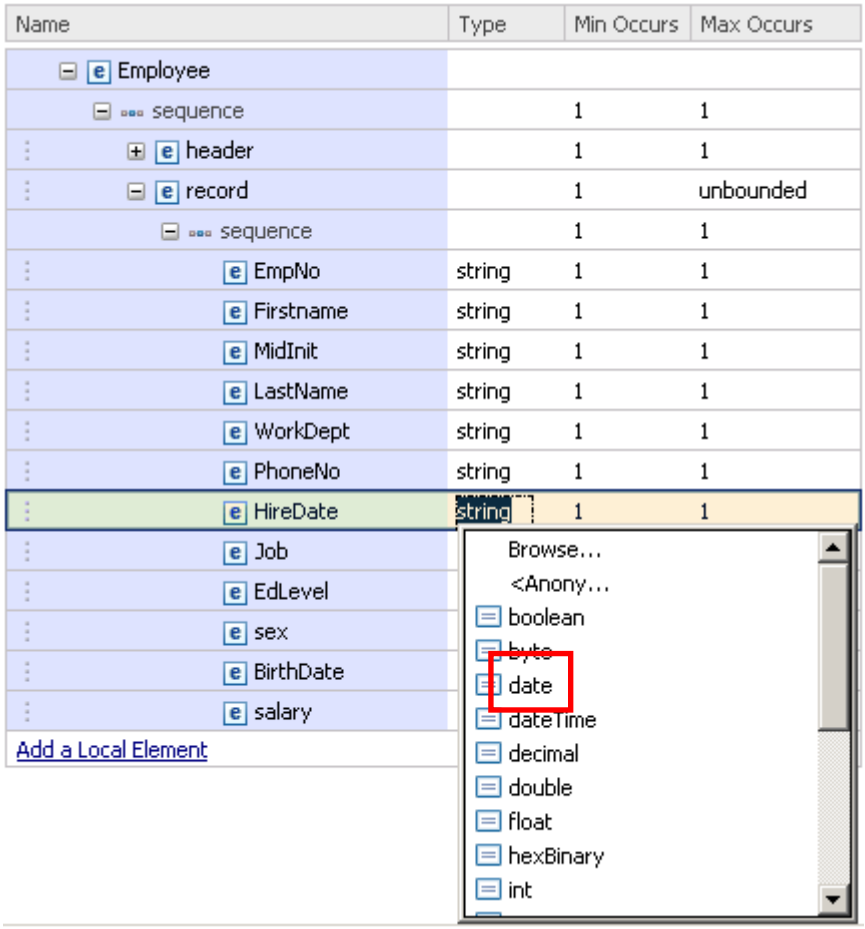
4. Change the field names to the following:

    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:

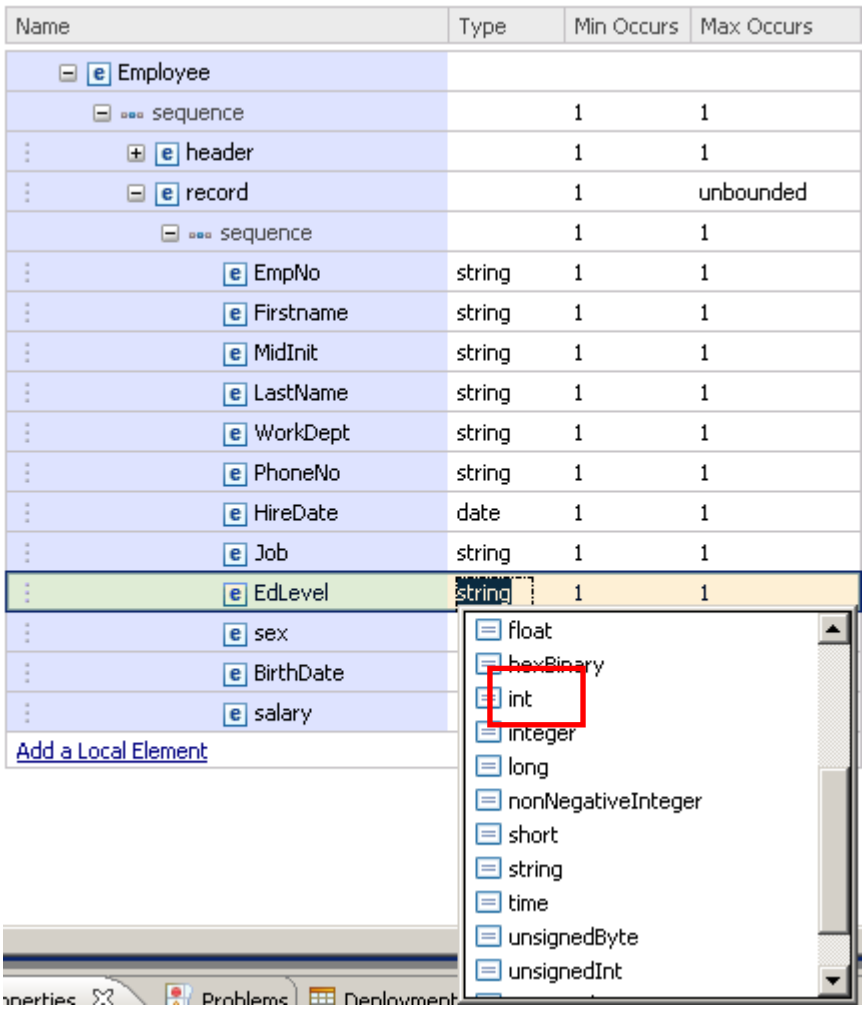


5. Now change the data type for some fields. Click in the Type column of the HireDate field and select "date" as the data type.

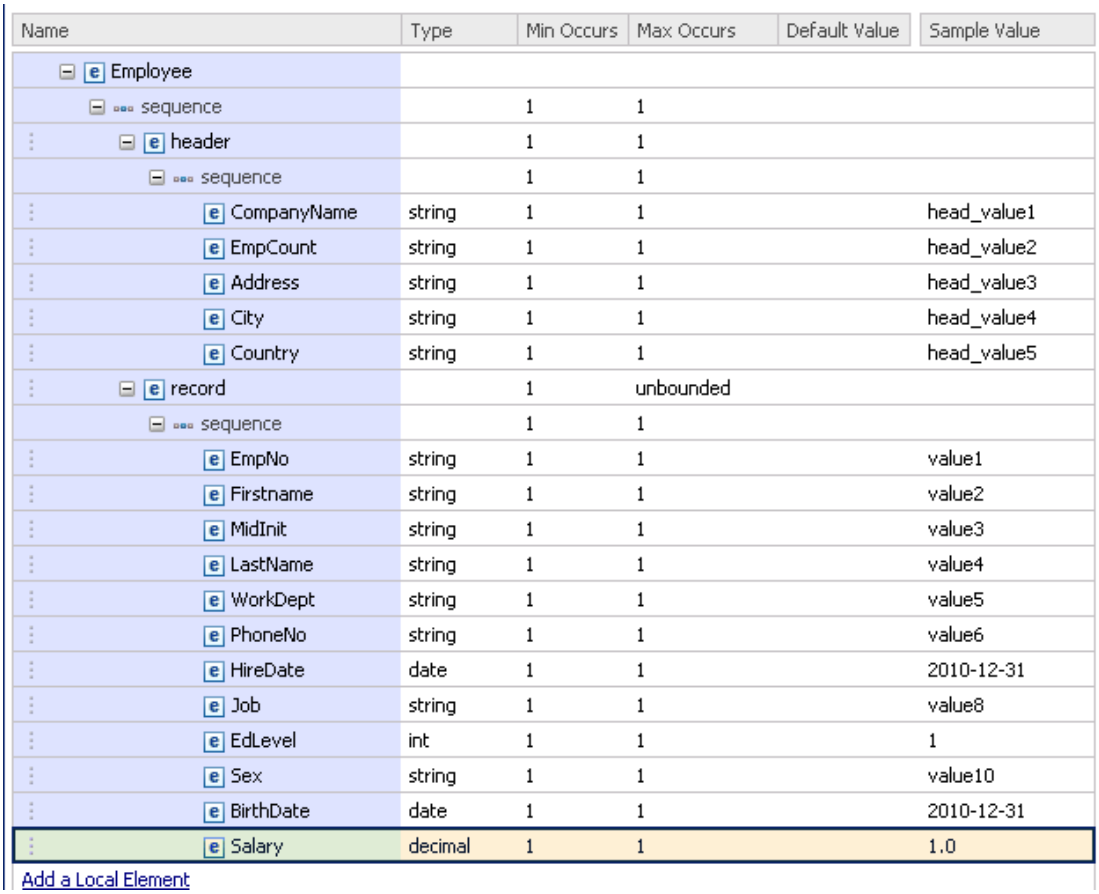


6. Repeat this step for BirthDate, setting type to "date".

   Change EdLevel's data type to "int".



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

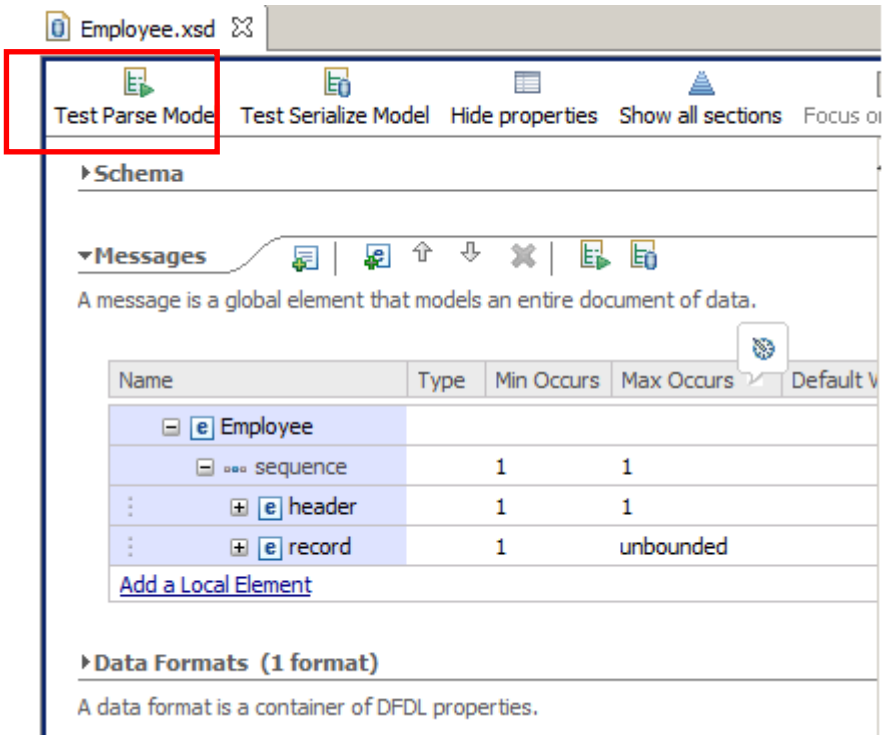
Add a Local Element

8. Save the Employee message model.

4. Test the Message Model

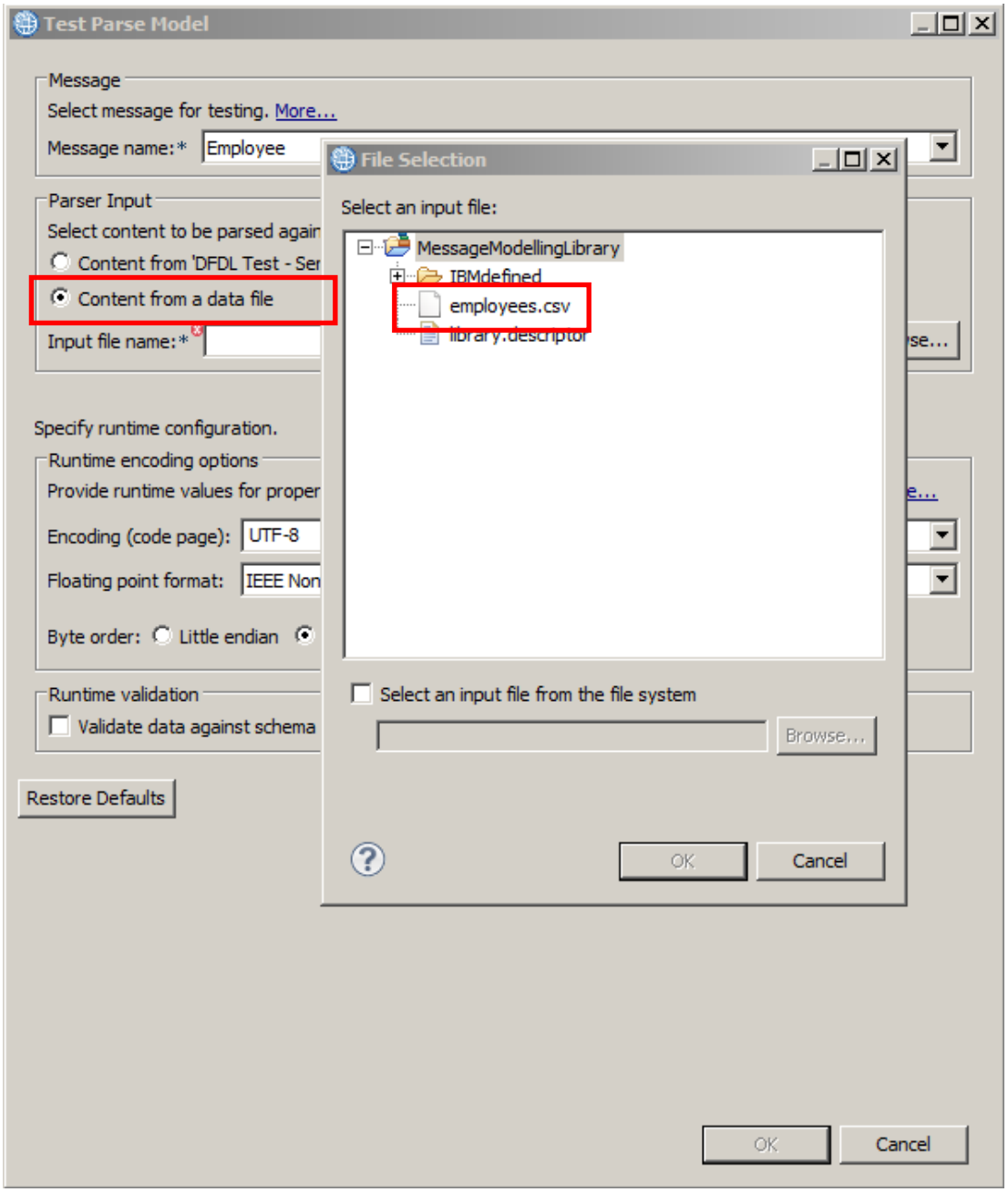
1. Now test the message model to validate that it parses the CSV data file correctly.

   Click the "Test Parse Model" icon.



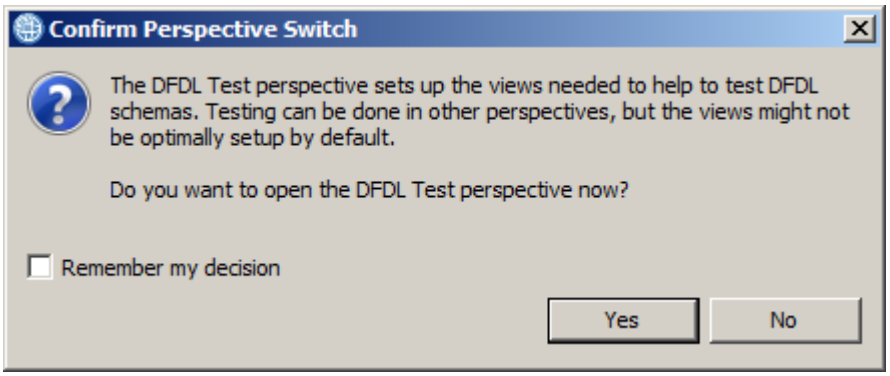
2. In the "Parse Input" section of the new window select the "Content from a data file" option and click the browse button to select the employees.csv file under MessageModellingLibrary.

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



   Then click OK in both windows.

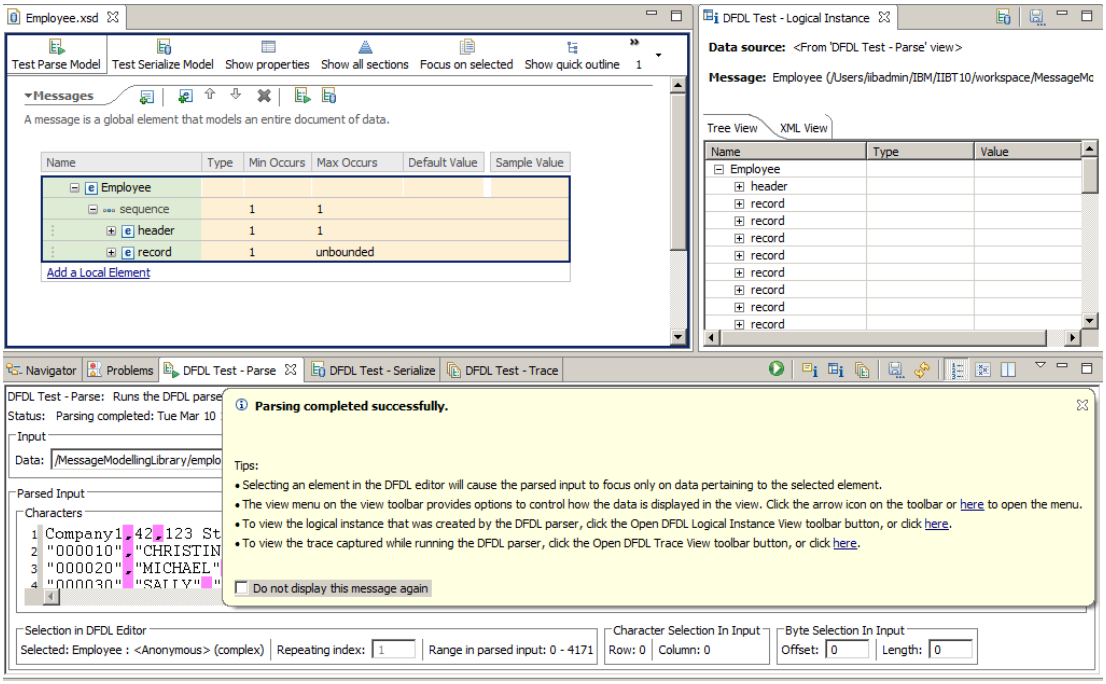
3. Confirm the popup by clicking yes, which will take you to the DFDL Test perspective



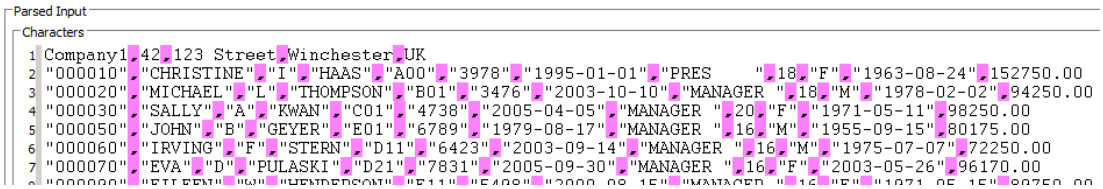
4. The DFDL Test perspective has several views:

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

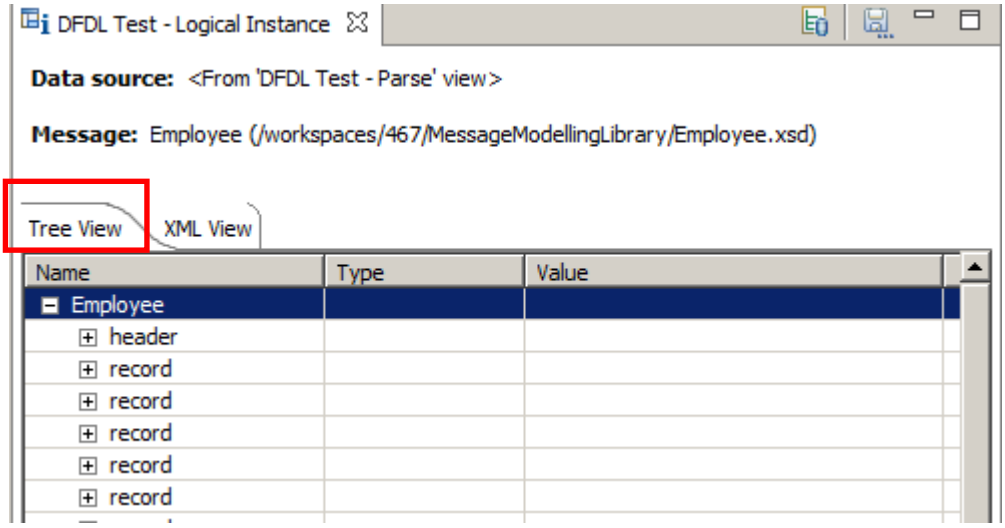
    When the perspective opens, you see a popup message with the parsing result, in this case successful. Close it by clicking in the "X".



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

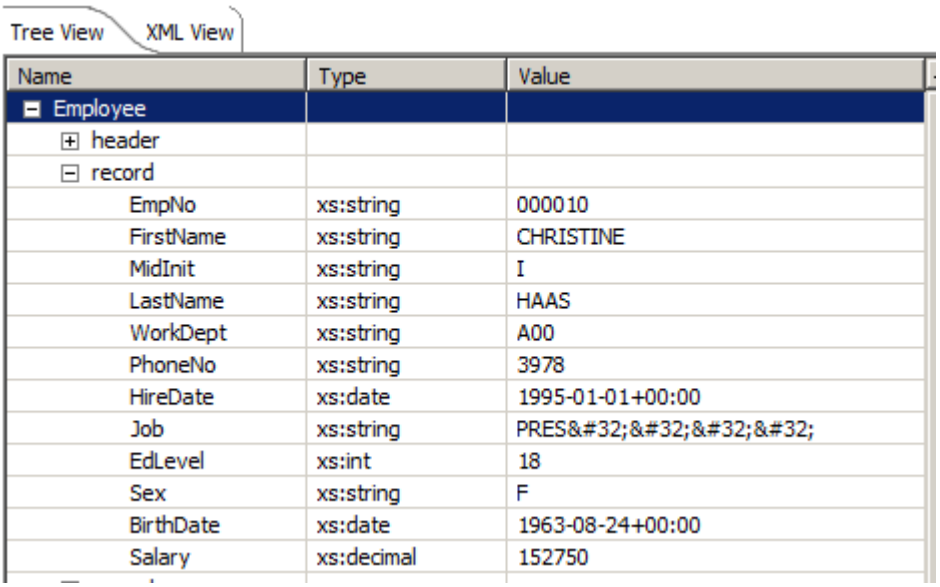


6. Look at the Logical instance view in the top right corner. Make sure the Tree View is selected; this shows the parsed message tree for the employees.csv file.

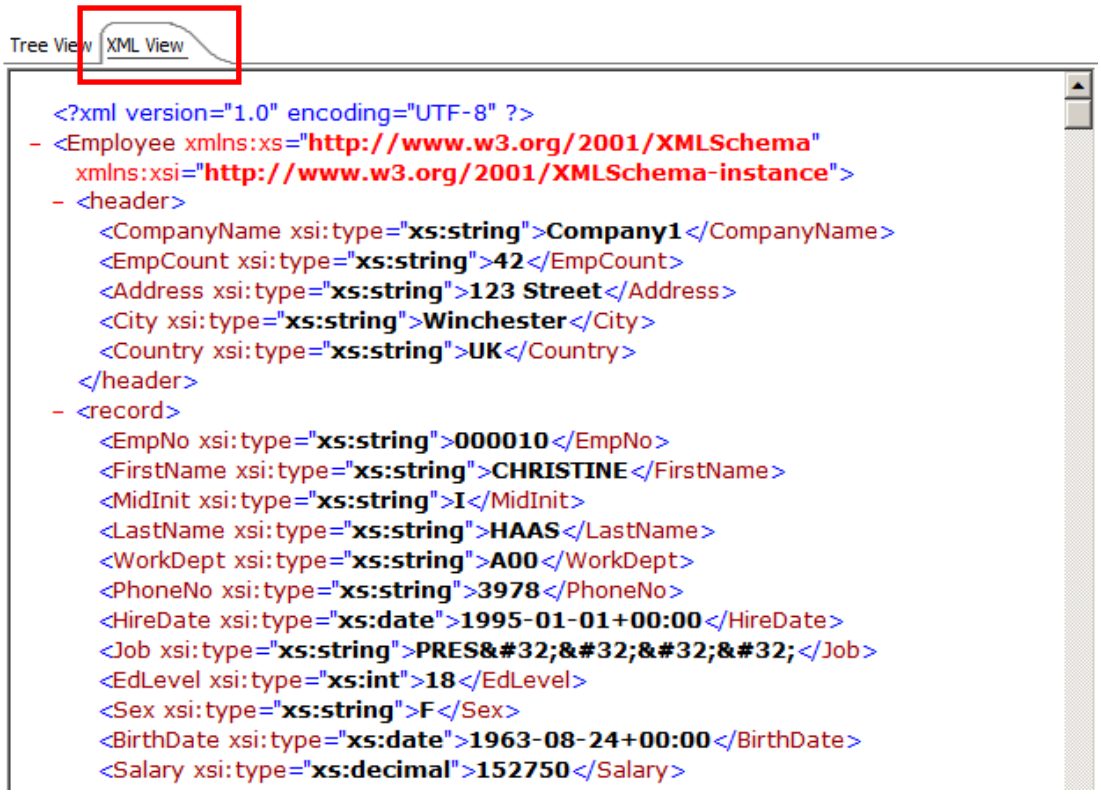


Expand a few records to check that the parsing was correct.

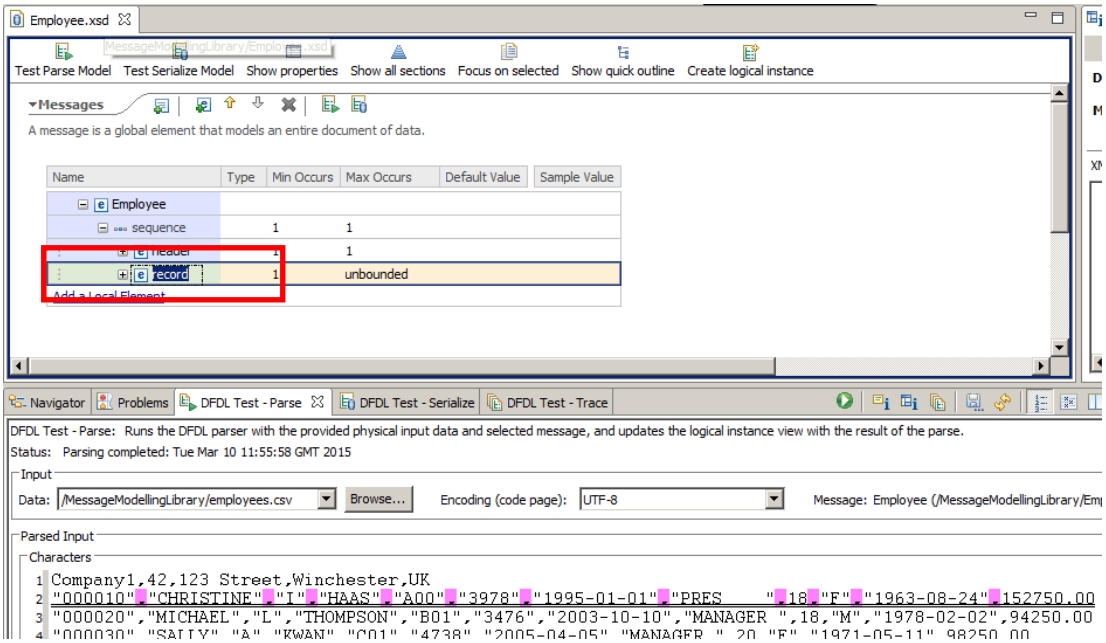
Note that the Logical Instance viewer shows elements of type "date" with a time-zone appended (+00.00 in this example), and also shows the full hex value of blank characters (in the Job element in this example).



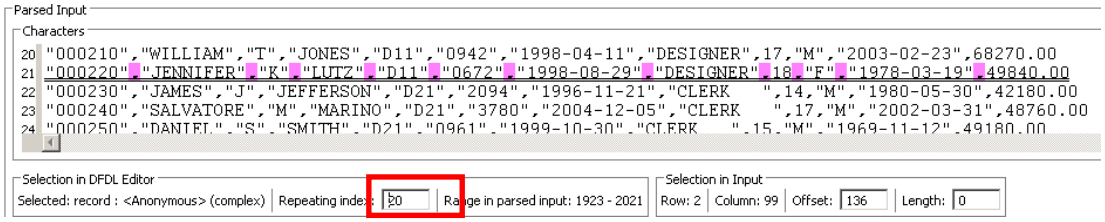
7. Click on XML View to show the same parsed records in XML format.



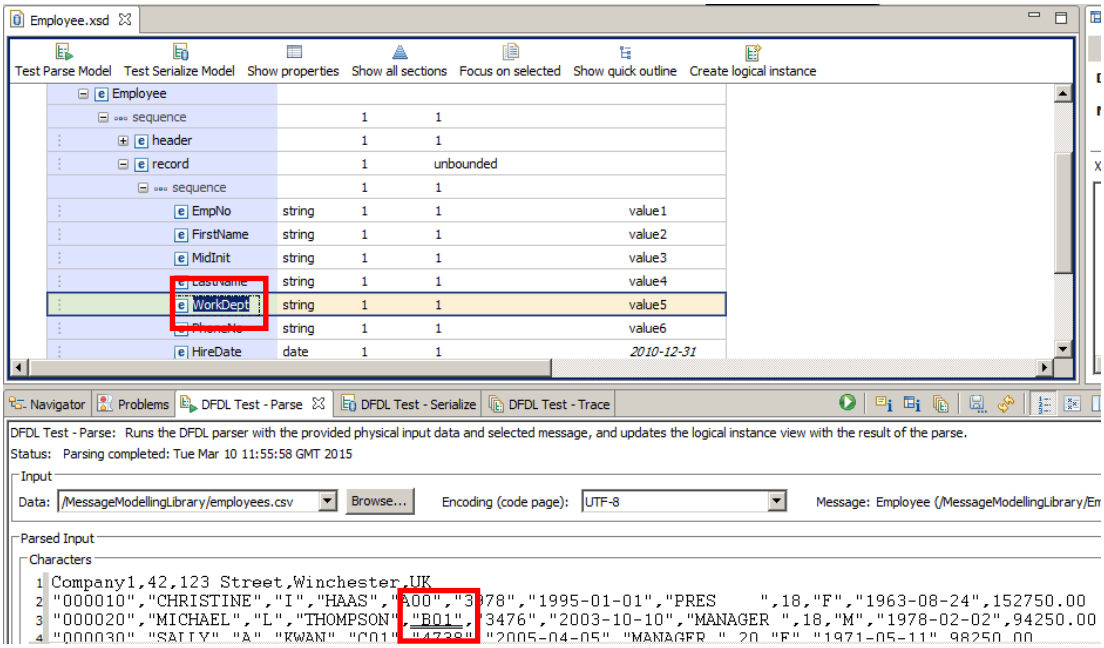
8. In the Employee message model, click on the record element. The first record in the input text will be underlined.



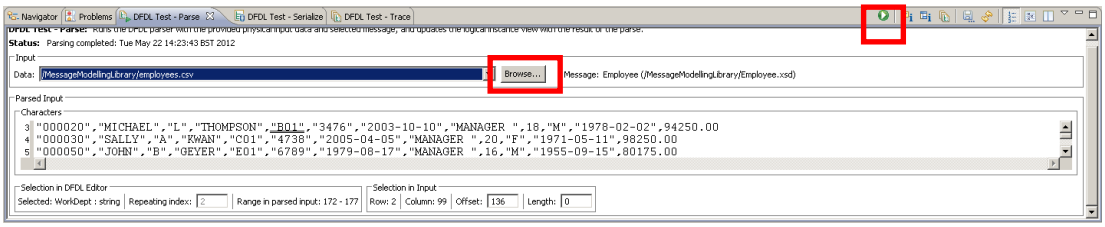
To go to another record in the input text, just change the "Repeating Index" number, and it will display the selected record.



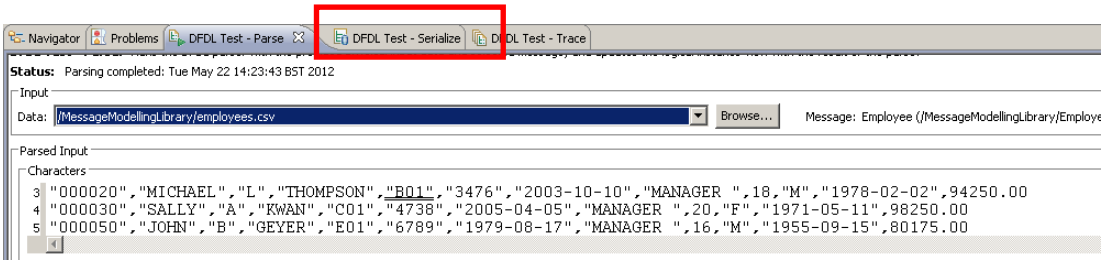
9. You can also click on any element in the message model to see where it's located in the input text.



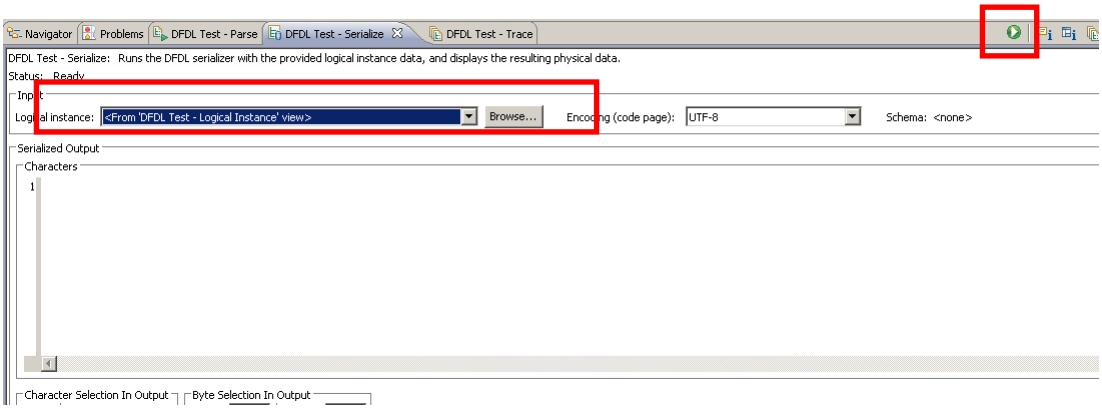
10. You could run the parser against other input text, by clicking the Browse button in the input section, selecting the file and clicking the Run Parser button.



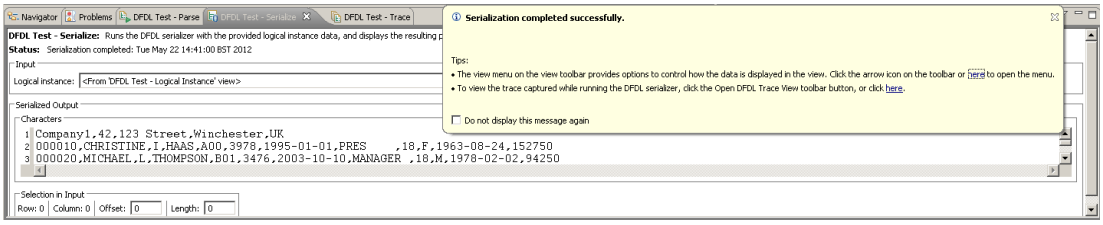
11. Now click on the “DFDL Test – Serialize” tab, next to the DFDL Test – Parse tab.



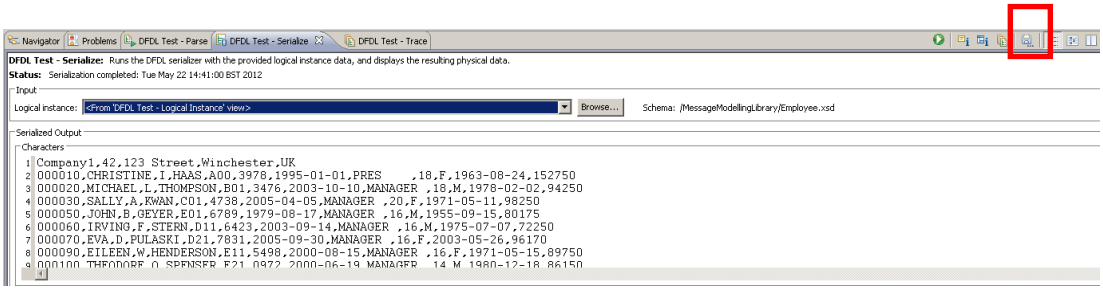
12. In the Input section, click on the combo box and select "<From 'DFDL Test - Logical Instance' view" and click the "Run Serializer" button (the icon on the top right of this view, as highlighted before).



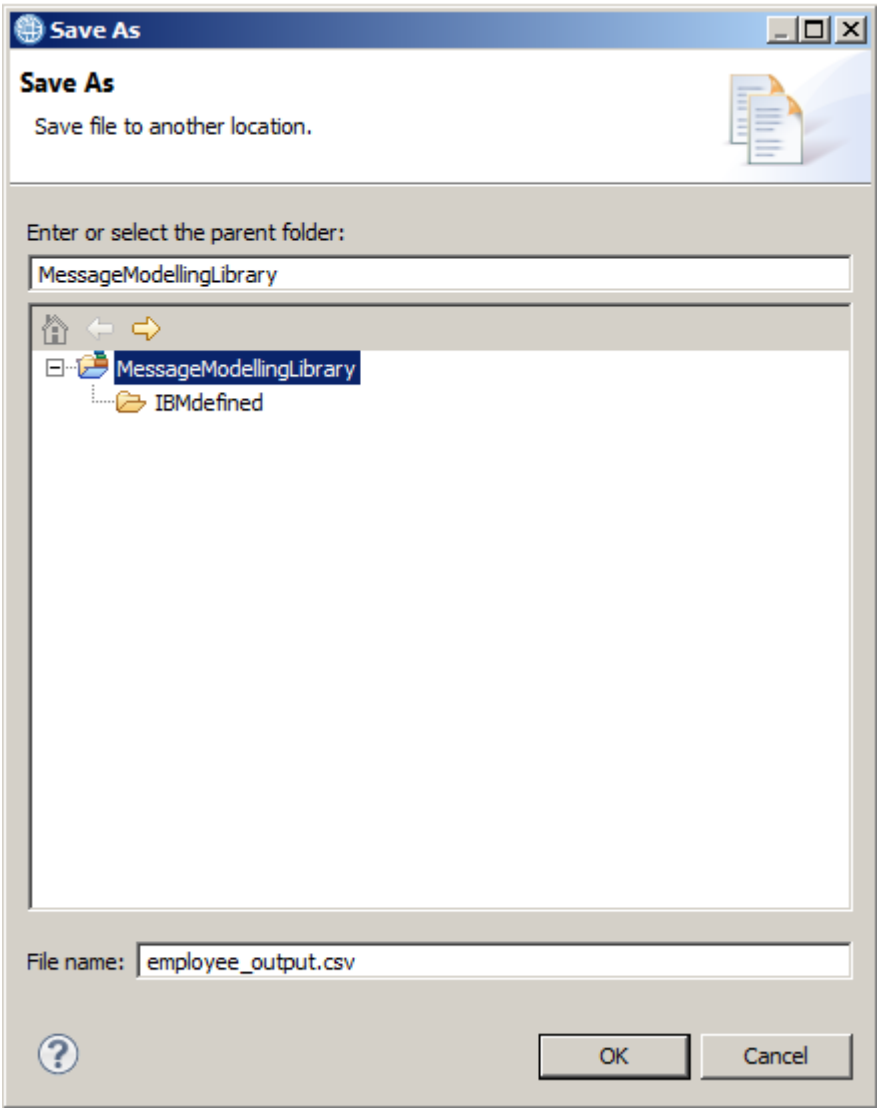
13. The Serializer has just created a text file from the message tree (previously parsed) using the current CSV message model:



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



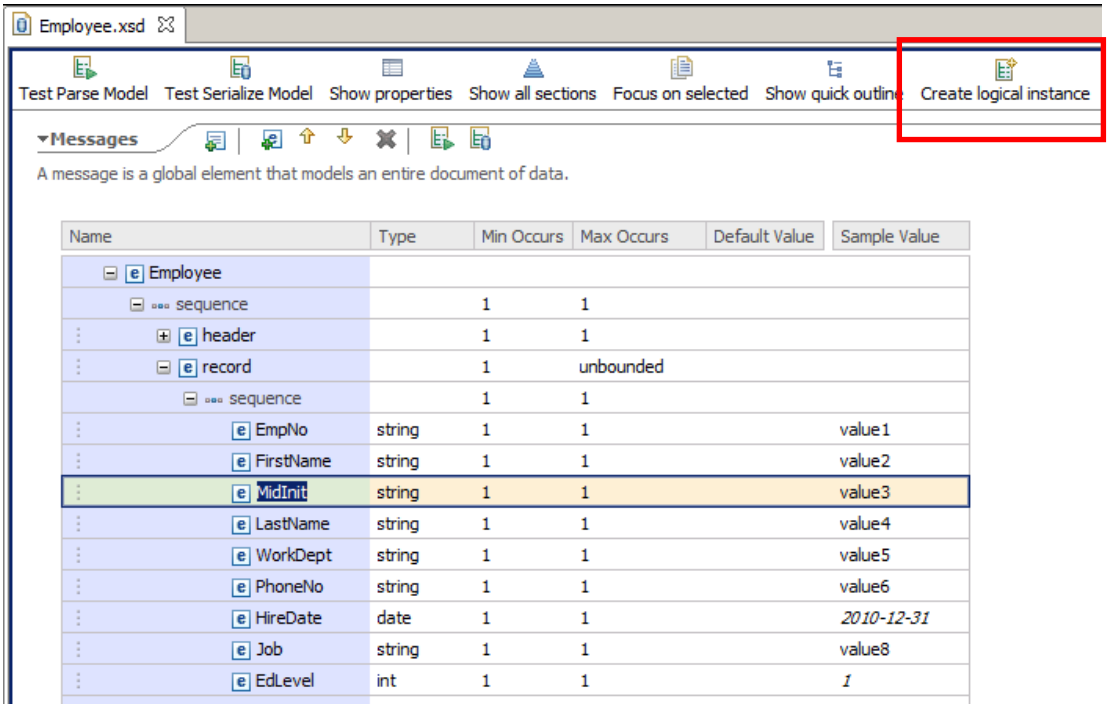
15. In the "Save as" window, select the "MessageModellingLibrary" folder, specify a name for the file (for example: "employee_output.csv") and click OK.



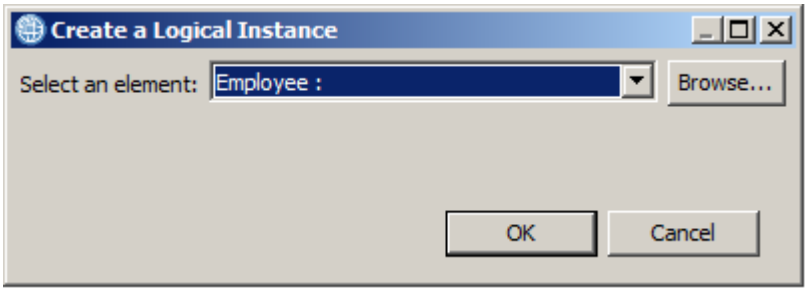
16. Now you are going to create a message tree using the "Sample Test Data" values and then serialize it to a file.

    Switch back to the Integration Development perspective.

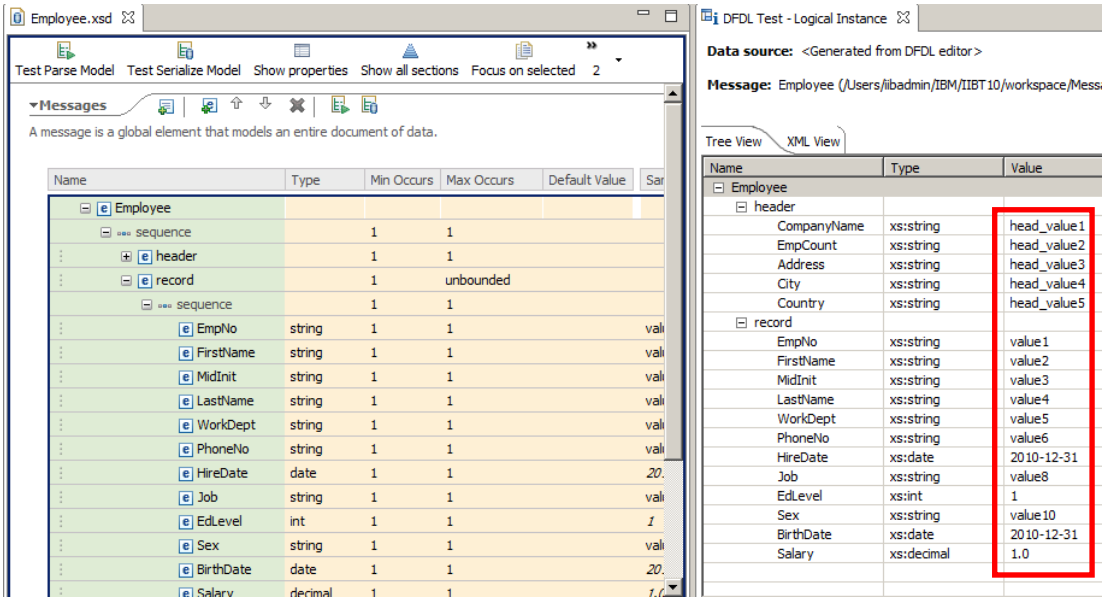
    Click on the "Create Logical Instance" icon. (You may need to move the slide-bar to the right to expose this icon).



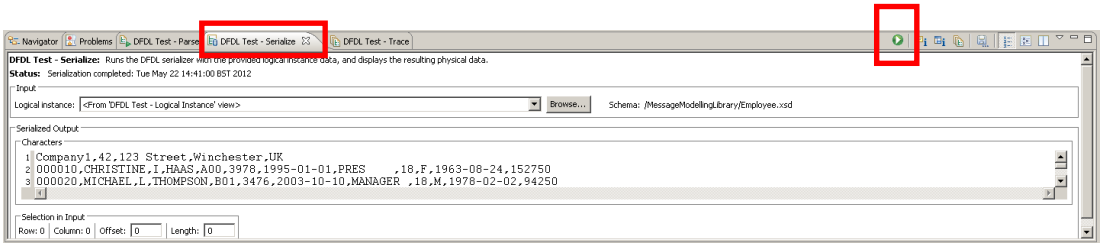
17. In the popup, check that "Employee:" is selected and click OK.



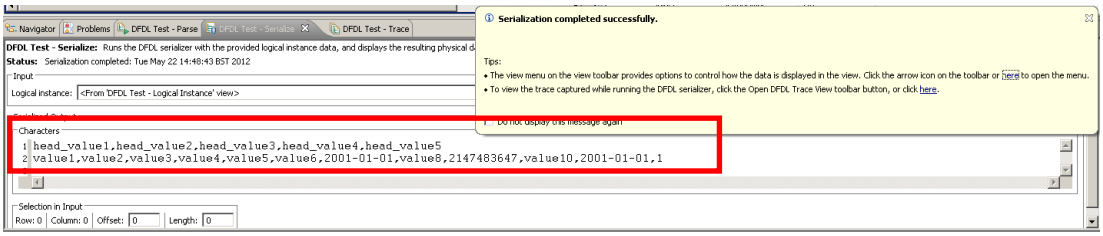
18. Look at the "DFDL Test - Logical Instance" view (in the right-hand pane), and inspect the message tree.



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.



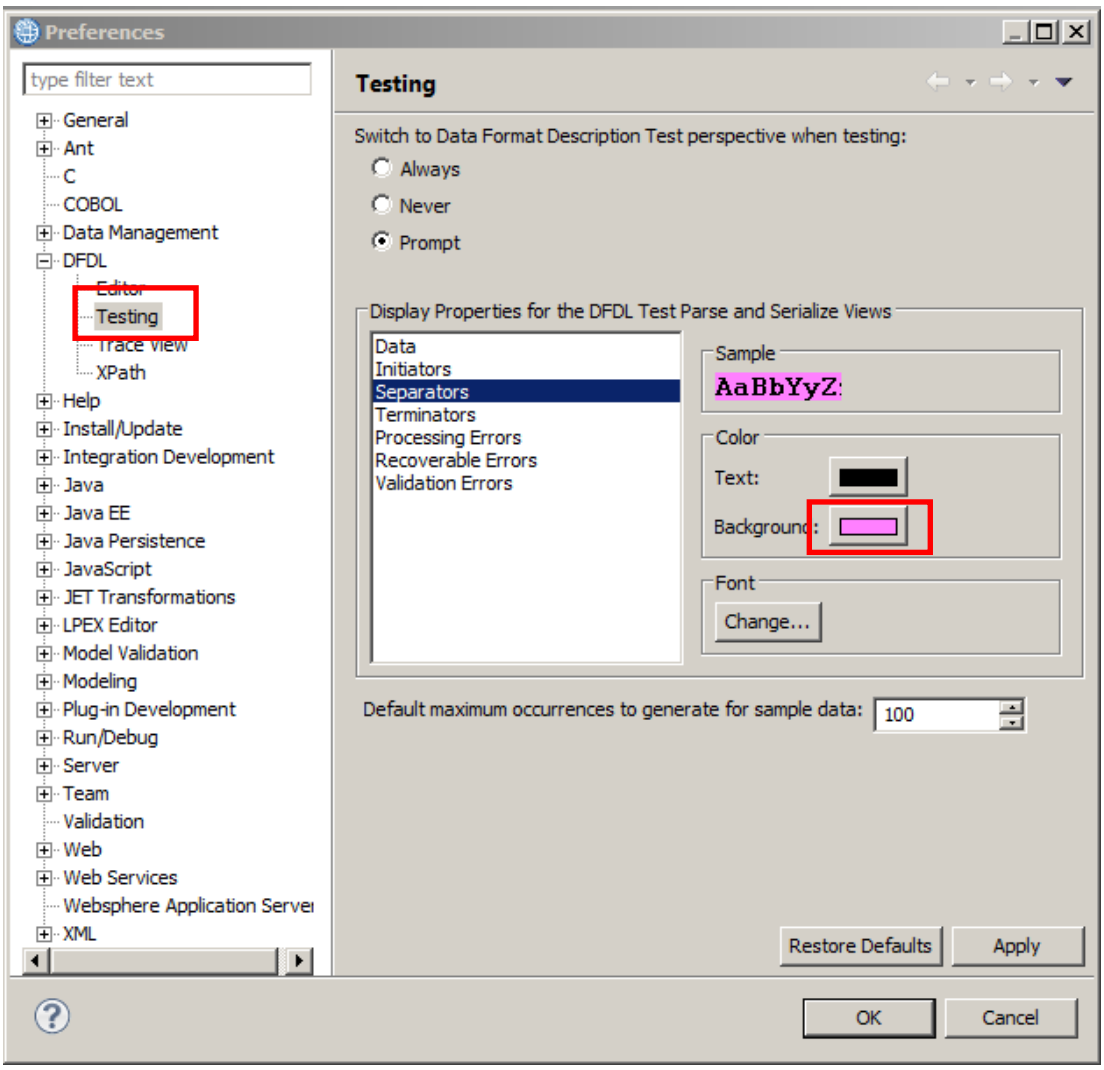
20. The Serializer has created a text file from the message tree generated using the Sample Test Data.



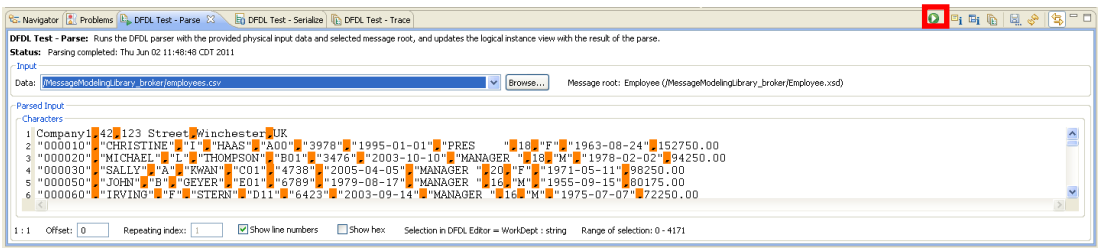
21. Finally, you can configure the DFDL Editor to change the Test Parse configuration properties.

    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.

END OF LAB GUIDE