Message Broker - Introduction to DFDL

Tim Kimber – WebSphere Message Broker Development
kimbert@uk.ibm.com

Agenda

- Introduction
- DFDL – a new standard for modeling text and binary data
- DFDL support in Message Broker
- Other enhancements
- Message Sets and MRM

Agenda

- Introduction
- DFDL – a new standard for modeling text and binary data
- DFDL support in Message Broker
- Other enhancements
- Message Sets and MRM

Introductions

- About me
  - Tim Kimber
  - Over 10 years working on parsers for WebSphere Message Broker
  - Worked on all aspects of MRM and XMLNSC parsers
  - Now lead developer for the new DFDL parser

- About the audience…
  - Experience with message broker
  - Experience with data formats

Agenda

- Introduction
- DFDL – a new standard for modeling text and binary data
- DFDL support in Message Broker
- Other enhancements
- Message Sets and MRM

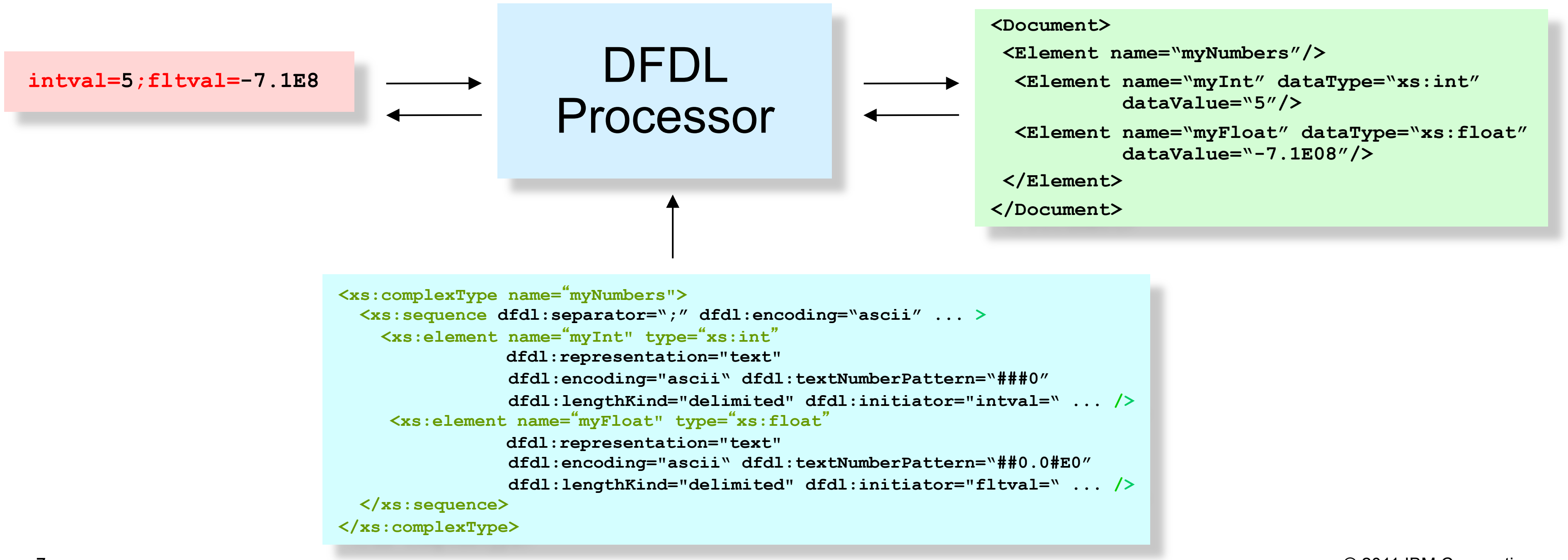
Why DFDL?

- Across organisations, the ability to exchange data is of vital importance
- But there has been no universal standard for modeling general text and binary data
  - XML -> use XML Schema
  - RDBMS -> use database schema
  - Text/binary -> ??
- Existing standards are too prescriptive: "*Put your data in this format!*"
- IBM products evolved their own way of modeling text and binary data based on customer need. Examples…
  - WebSphere Message Broker: MRM message set
  - WebSphere ESB, IBM Process Server: Data Handlers
  - Transformation Extender: Type Trees
  - DataPower: FFD
  - Cast Iron: Flat File Schema
  - Sterling Integrator: DDF and IDF files

✓ ***DFDL:*** *a universal, shareable, non-prescriptive description for general text & binary data formats*

DFDL processor – Do something useful with a DFDL schema

- A DFDL processor uses a DFDL schema to understand a data stream
- It consists of a DFDL parser and (optionally) a DFDL unparser
- The DFDL parser reads a data stream and creates a DFDL ‘info set’
- The DFDL unparser takes a DFDL ‘info set’ and writes a data stream

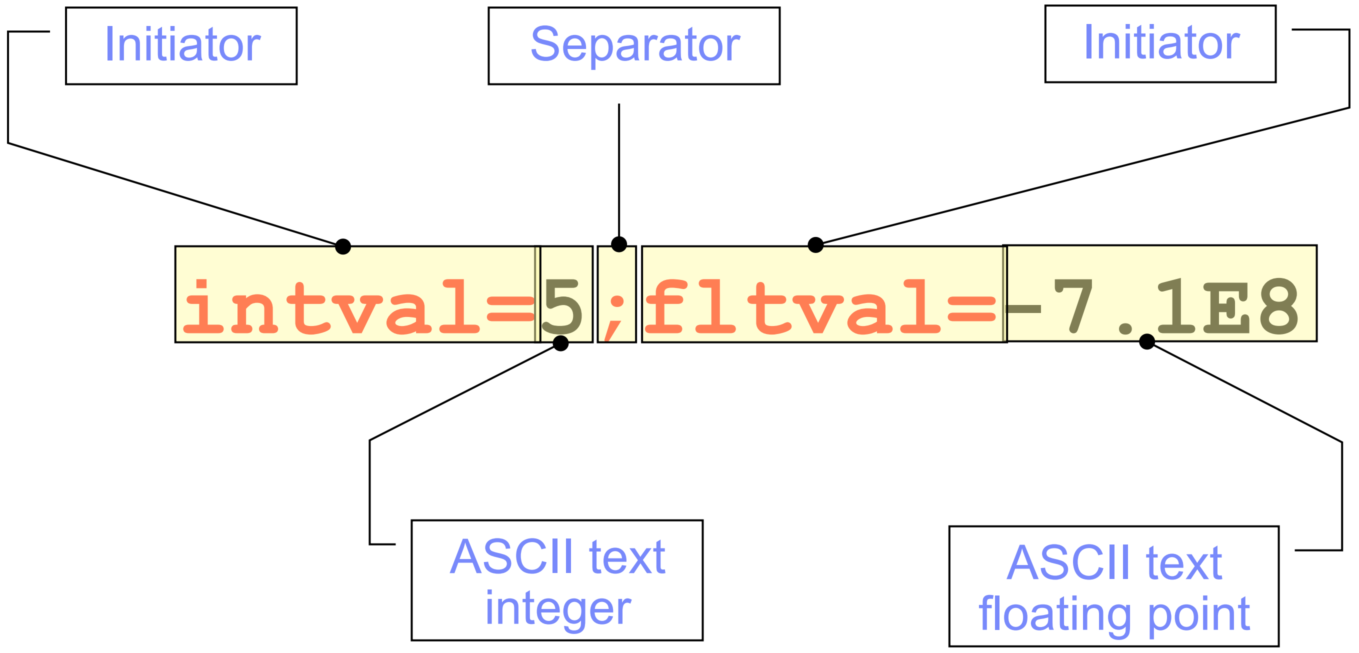


```
intval=5;fltval=-7.1E8
```

```
<Document>
 <Element name="myNumbers"/>
  <Element name="myInt" dataType="xs:int"
           dataValue="5"/>
  <Element name="myFloat" dataType="xs:float"
           dataValue="-7.1E08"/>
 </Element>
</Document>
```

```
<xs:complexType name="myNumbers">
  <xs:sequence dfdl:separator=";" dfdl:encoding="ascii" ... >
    <xs:element name="myInt" type="xs:int"
                dfdl:representation="text"
                dfdl:encoding="ascii" dfdl:textNumberPattern="###0"
                dfdl:lengthKind="delimited" dfdl:initiator="intval=" ... />
     <xs:element name="myFloat" type="xs:float"
                dfdl:representation="text"
                dfdl:encoding="ascii" dfdl:textNumberPattern="##0.0#E0"
                dfdl:lengthKind="delimited" dfdl:initiator="fltval=" ... />
  </xs:sequence>
</xs:complexType>
```

Example – Sequence group containing two fields



Separators, initiators (aka tags), & terminators are all examples in DFDL of *delimiters*

Example DFDL xsd with annotations

```
intval=5;fltval=-7.1E8
```

```
<xs:complexType name="myNumbers">
  <xs:sequence>
    <xs:annotation>
      <xs:appinfo source="http://www.ogf.org/dfdl/v1.0">
        <dfdl:sequence separator=";" encoding="ascii"/>
      </xs:appinfo>
    </xs:annotation>
    <xs:element name="myInt" type="xs:int">
      <xs:annotation>
        <xs:appinfo source="http://www.ogf.org/dfdl/v1.0">
           <dfdl:element representation="text"
             encoding="ascii" textNumberPattern="###0"
             lengthKind="delimited" initiator="intval=" .../>
        </xs:appinfo>
      </xs:annotation>
   </xs:element>
    <xs:element name="myFloat" type="xs:float">
      <xs:annotation>
        <xs:appinfo source="http://www.ogf.org/dfdl/v1.0">
           <dfdl:element representation="text"
             encoding="ascii" textNumberPattern="##0.0#E0"
             lengthKind="delimited" initiator="fltval=" .../>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

DFDL is...Powerful and Flexible

- DFDL is not a data format
  - it is a language for describing data formats
- DFDL can describe almost any data format
  - Textual and binary
  - Commercial record-oriented
  - Scientific and numeric
  - Modern and legacy
  - Industry standards
- DFDL is a powerful language, supporting
  - Text, binary and bit data.
  - Fixed-length, delimited, pattern and prefixed lengths
  - Nil values for out-of-range values
  - Default values
  - Validation to XML Schema 1.0 rules
  - …and a lot more as well
    - See reference slides for more details

DFDL is...Open

- A DFDL data format
  - Is portable
    - Can be processed by any DFDL processor that implements the DFDL specification.
  - Is editable using any text editor
    - although IBM has developed a good graphical editor/debugger.
  - Can be generated by a script
    - the DFDL annotation language is published, and is not a trade secret
    - useful when modeling large data industry standard formats
- DFDL is an open standard
  - Hosted by the Open Grid Forum (OGF)
    - http://www.ogf.org/
  - Currently at version 1.0
    - ‘Proposed Recommendation’ status

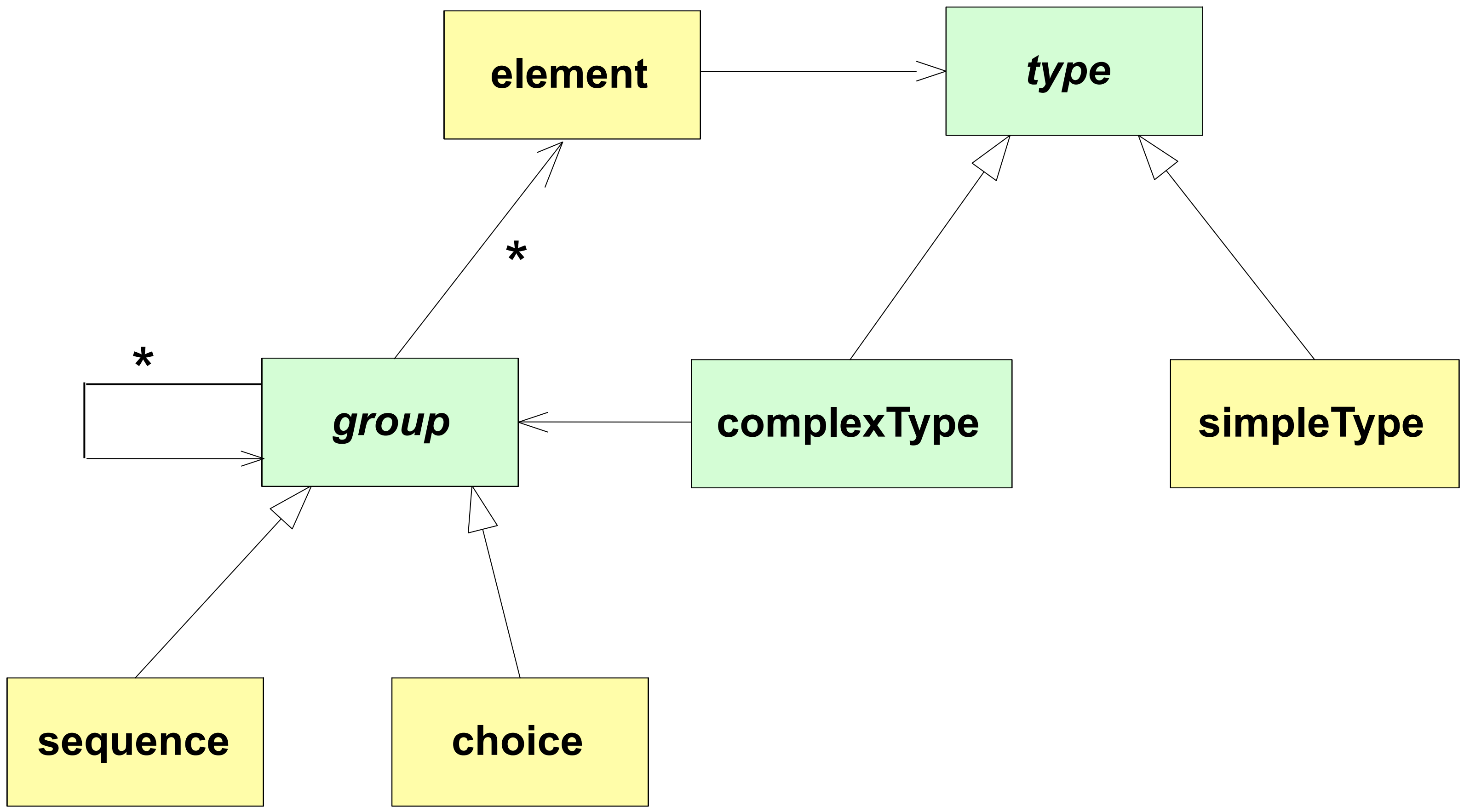
DFDL is...Based on XML Schema

- XML Schema is the most common way to describe tree-structured data
  - So DFDL uses it to describe the logical structures and types in a document
- … but XML Schema can only describe one data format (XML)
  - So DFDL adds XML Schema annotations that describe the physical data format
- XML Schema describes a lot of XML-specific things
  - So DFDL uses a tight subset of XML Schema
    - Element names and namespaces
    - Complex type and groups
    - Simple types and their facets (constraints)
  - Enough to do what DFDL requires, and no more.
- Note : Annotations are defined in the XML Schema specification
  - So a DFDL xsd is a standard xsd.
  - DFDL annotations do not interfere with the normal usage of an xsd
  - They are ignored by all except DFDL processors

DFDL is...Smart

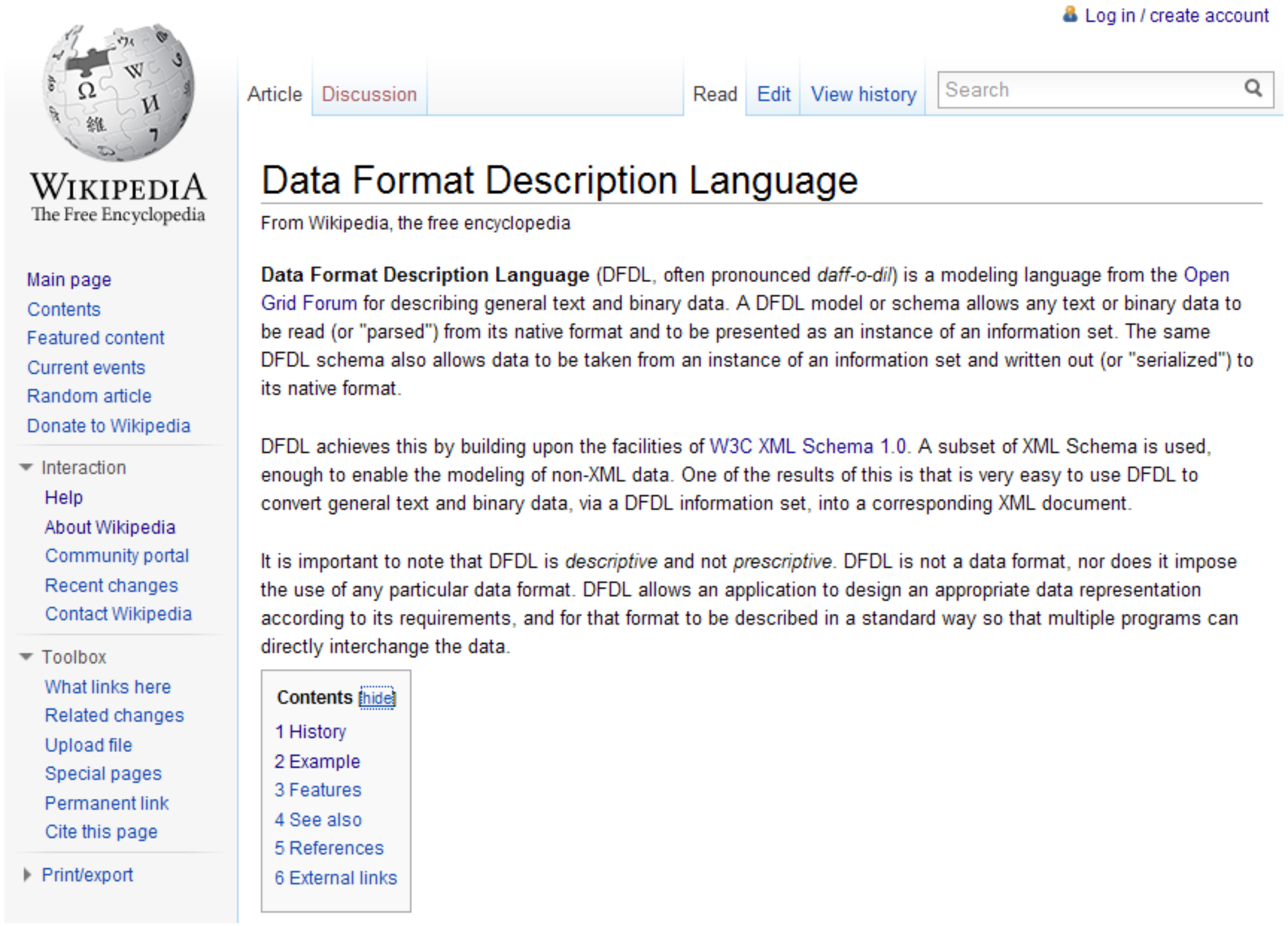
- DFDL will automatically resolve choices and optional elements
  - By attempting each available path through the model
  - If a path fails, DFDL will backtrack and try another path
    - Unless the model tells it not to do that
- DFDL has an expression language for
  - Specifying property values
    - `../repeatCount` is the number of occurrences of this element
    - `/delim` is the delimiter for this structure
  - Controlling the parser
    - `"This is the correct path if the current field is the same as ../header/id"`
    - `"Report an error if this field value is less than /header/maxValue"`
- DFDL works in both directions
  - It can parse a document to produce a tree of information (info set)
  - It can serialize an info set to produce a document
  - The same DFDL schema can be used for both operations

DFDL language - objects



DFDL properties are placed on yellow objects only

Getting started with DFDL



Data Format Description Language

From Wikipedia, the free encyclopedia

Data Format Description Language (DFDL, often pronounced *daff-o-dil*) is a modeling language from the Open Grid Forum for describing general text and binary data. A DFDL model or schema allows any text or binary data to be read (or "parsed") from its native format and to be presented as an instance of an information set. The same DFDL schema also allows data to be taken from an instance of an information set and written out (or "serialized") to its native format.

DFDL achieves this by building upon the facilities of W3C XML Schema 1.0. A subset of XML Schema is used, enough to enable the modeling of non-XML data. One of the results of this is that is very easy to use DFDL to convert general text and binary data, via a DFDL information set, into a corresponding XML document.

It is important to note that DFDL is *descriptive* and not *prescriptive*. DFDL is not a data format, nor does it impose the use of any particular data format. DFDL allows an application to design an appropriate data representation according to its requirements, and for that format to be described in a standard way so that multiple programs can directly interchange the data.

Contents [hide]

DFDL features

- Text data types such as strings, numbers, zoned decimals, calendars, booleans
- Binary data types such as integers, floats, BCD, packed decimals, calendars, booleans
- Fixed length data and data delimited by text or binary markup
- Language data structures found in COBOL, C and PL/1
- Industry standards such as SWIFT, HL7, FIX, HIPAA, X12, EDIFACT, ISO8583
- Bi-directional text
- Bit data of arbitrary length
- Pattern languages for text numbers and calendars
- Ordered, unordered and floating content
- Default values on parsing and serializing
- Nil values for handling out-of-band data
- Fixed and variable arrays
- XPath 2.0 expression language including variables to model dynamic data
- Speculative parsing to resolve choices and optional content
- Validation to XML Schema 1.0 rules
- Scoping mechanism to allow common property values to be applied at multiple points
- Hide elements in the data
- Calculate element values

Example – DFDL schema (short form)

```
<xs:complexType name=“myNumbers">
  <xs:sequence dfdl:separator=“;” dfdl:encoding=“ascii” >
    <xs:element name=“myInt" type=“xs:int”
                          dfdl:representation="text"
                         dfdl:encoding=“ascii" dfdl:textNumberPattern=“###0”
                         dfdl:lengthKind="delimited" dfdl:initiator="intval=" … />
    <xs:element name=“myFloat" type=“xs:float”
                          dfdl:representation="text"
                         dfdl:encoding=“ascii" dfdl:textNumberPattern=“##0.0#E0”
                         dfdl:lengthKind="delimited" dfdl:initiator="fltval=" … />
  </xs:sequence>
</xs:complexType>
```

DFDL
properties

DFDL language – basic annotations

Annotation	***Used on Component***	***Purpose***
dfdl:element	xs:element xs:element reference	Contains the DFDL properties of an xs:element and xs:element reference
dfdl:choice	xs:choice	Contains the DFDL properties of an xs:choice.
dfdl:sequence	xs:sequence	Contains the DFDL properties of an xs:sequence.
dfdl:group	xs:group reference	Contains the DFDL properties of an xs:group reference to a group definition containing an xs:sequence or xs:choice.
dfdl:simpleType	xs:simpleType	Contains the DFDL properties of an xs:simpleType
dfdl:format	xs:schema dfdl:defineFormat	Contains a set of DFDL properties that can be used by multiple DFDL schema components. When used directly on xs:schema, the property values act as defaults for all components in the DFDL schema.
dfdl:defineFormat	xs:schema	Defines a reusable data format by associating a name with a set of DFDL properties contained within a child dfdl:format annotation. The name can be referenced from DFDL annotations on multiple DFDL schema components, using dfdl:ref.

DFDL language – more advanced annotations

Annotation	***Used on Component***	***Purpose***
dfdl:assert	xs:element, xs:choice xs:sequence, xs:group	Defines a test to be used to ensure the data are well formed. Used only when parsing data.
dfdl:discriminator	xs:element, xs:choice xs:sequence, xs:group	Defines a test to be used when resolving a point of uncertainty such as choice branches or optional elements. Used only when parsing.
dfdl:escapeScheme	dfdl:defineEscapeScheme	Defines a scheme by which quotation marks and escape characters can be specified. This is for use with delimited text formats.
dfdl:defineEscapeScheme	xs:schema	Defines a named, reusable escape scheme. The name can be referenced from DFDL annotations on multiple DFDL schema components.
dfdl:hidden	xs:sequence	Defines a hidden element that appears in the schema for use by the DFDL processor, but is not part of the infoset.
dfdl:defineVariable	xs:schema	Defines a variable that can be referenced elsewhere. This can be used to communicate a parameter from one part of processing to another part.
dfdl:newVariableInstance	xs:element, xs:choice xs:sequence, xs:group	Creates a new instance of a variable
dfdl:setVariable	xs:element, xs:choice xs:sequence, xs:group	Sets the value of a variable whose declaration is in scope

Links

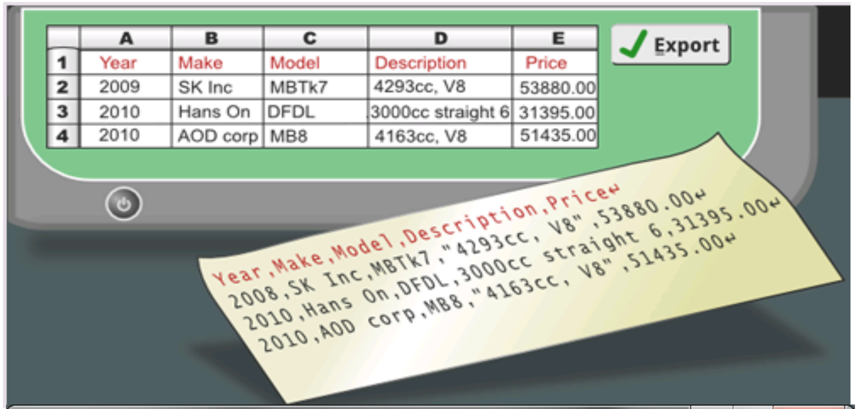
- OGF DFDL home page: http://www.ogf.org/dfdl/
- DFDL 1.0 specification (pdf): http://www.ogf.org/documents/GFD.174.pdf
- DFDL 1.0 specification (html): http://www.ogf.org/dfdl/spec.php
- DFDL tutorial: http://forge.gridforum.org/sf/docman/do/listDocuments/projects.dfdl-wg/docman.root.current_0.documents
- DFDL-WG GridForge project: https://forge.ogf.org/sf/projects/dfdl-wg
- DFDL Wikipedia page: http://en.wikipedia.org/wiki/DFDL

Agenda

- DFDL – a new standard for modeling text and binary data
- DFDL support in Message Broker
- Other enhancements
- Message Sets and MRM

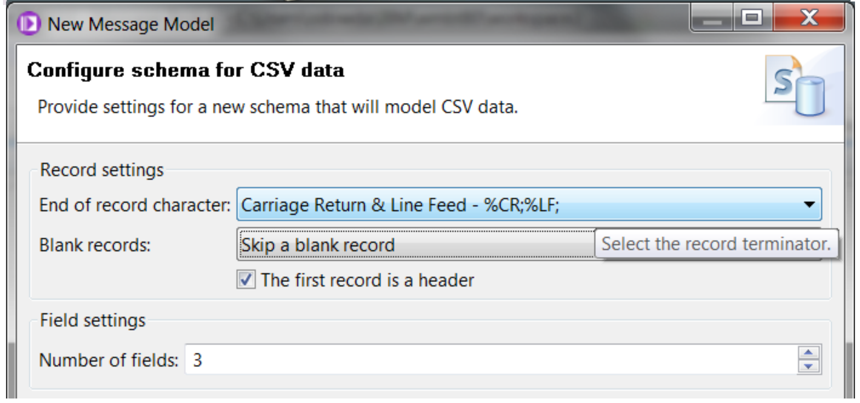
DFDL support in runtime and toolkit

- MB uses IBM DFDL 1.0.2 component
- DFDL models
  - Schema files reside in MB libraries
  - Not in Message Sets
- DFDL domain and high-performance parser
  - Available in nodes, ESQL, Java, …
  - On-demand, streaming parser
  - Use instead of MRM CWF/TDS
- Tooling for creating DFDL models in MBTK
  - Guided authoring wizards
  - COBOL importer
  - DFDL graphical editor
- DFDL model debugger
  - Debug parsing & writing of data in MBTK
  - No deploy necessary!
- DFDL schema deployed to broker in BAR file
  - No dictionary file!



	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

```
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
```



New Message Model

Configure schema for CSV data

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

Record settings

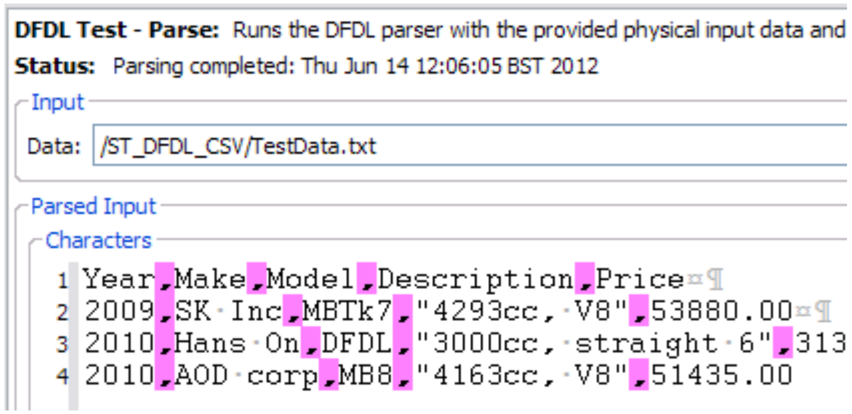
End of record character: Carriage Return & Line Feed - %CR;%LF;

Blank records: Skip a blank record — Select the record terminator.

☑ The first record is a header

Field settings

Number of fields: 3



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

Status: Parsing completed: Thu Jun 14 12:06:05 BST 2012

Input

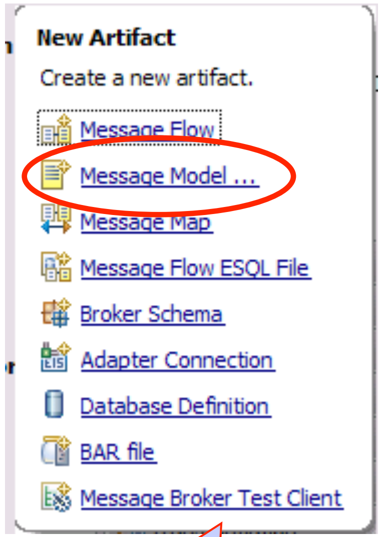
Data: /ST_DFDL_CSV/TestData.txt

Parsed Input

Characters

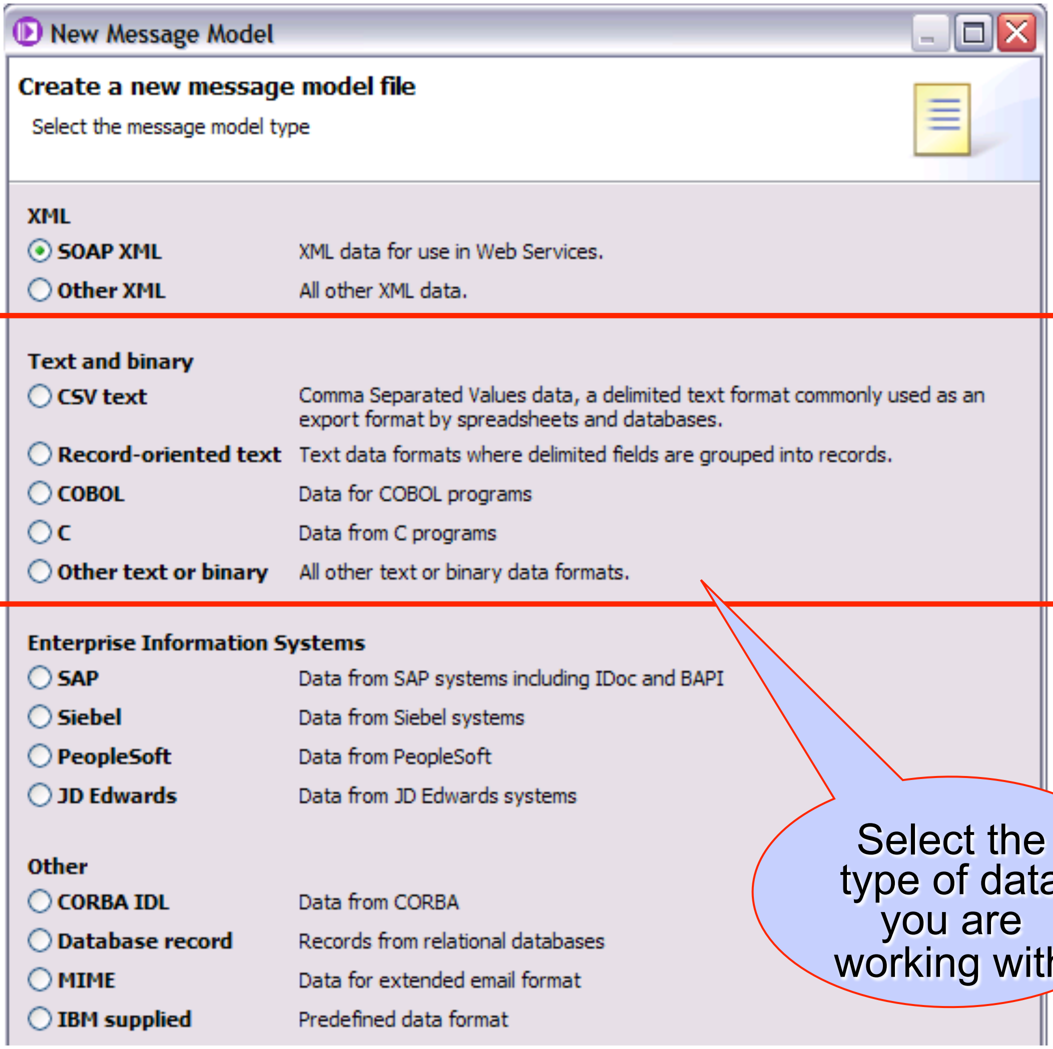
```
Year,Make,Model,Description,Price
2009,SK Inc,MBTk7,"4293cc, V8",53880.00
2010,Hans On,DFDL,"3000cc, straight 6",313
2010,AOD corp,MB8,"4163cc, V8",51435.00
```

Creating a DFDL model


New Artifact

Create a new artifact.

- Message Flow
- Message Model ...
- Message Map
- Message Flow ESQL File
- Broker Schema
- Adapter Connection
- Database Definition
- BAR file
- Message Broker Test Client


New Message Model

Create a new message model file

Select the message model type

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 from 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

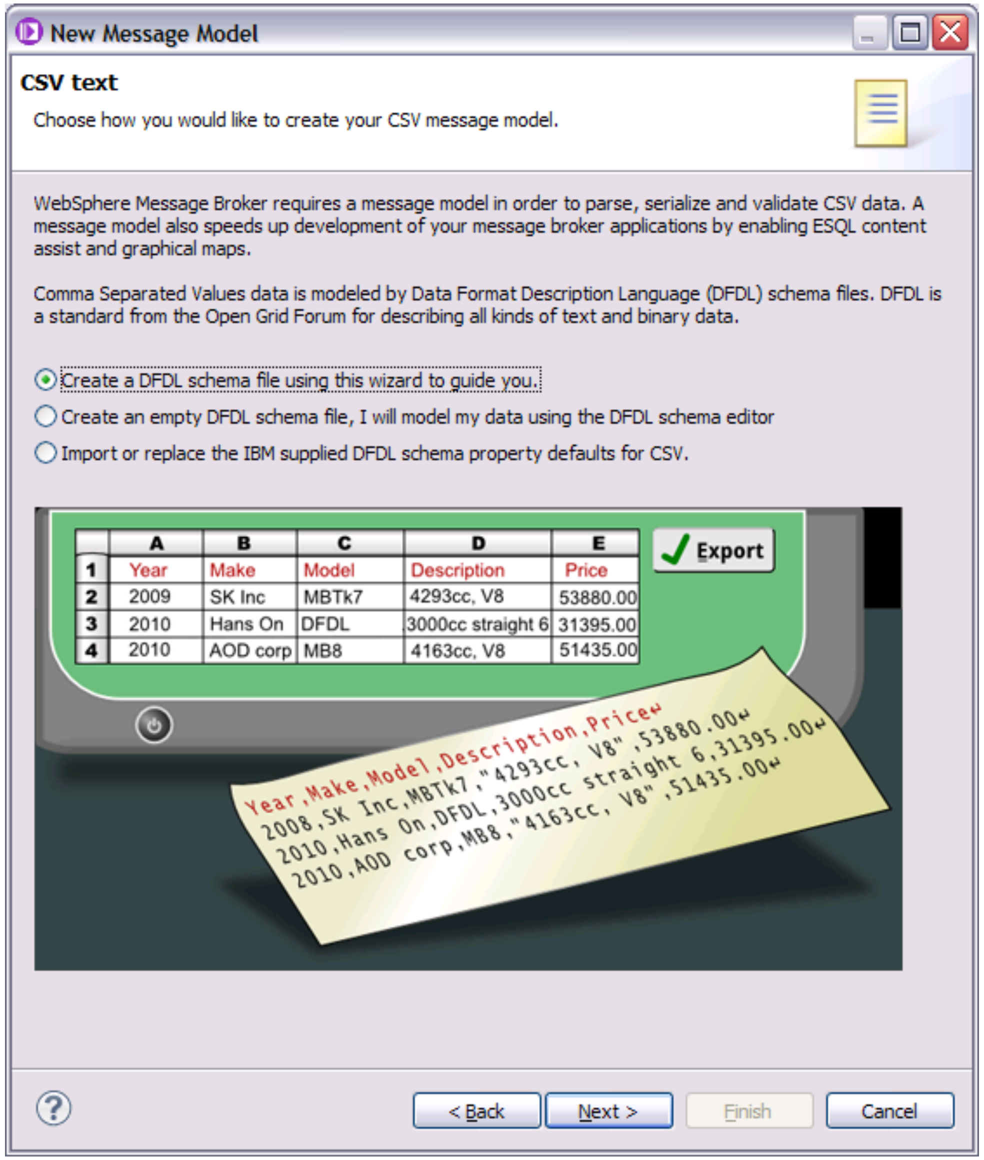
CORBA IDL	Data from CORBA
Database record	Records from relational databases
MIME	Data for extended email format
IBM supplied	Predefined data format

- Or you can drop an existing DFDL schema into a Message Broker library

Wizard options for creating a DFDL model

- Guided authoring
- Using the DFDL editor
- Importing from other metadata
- Already have DFDL schema

Select how you want to create the DFDL 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.

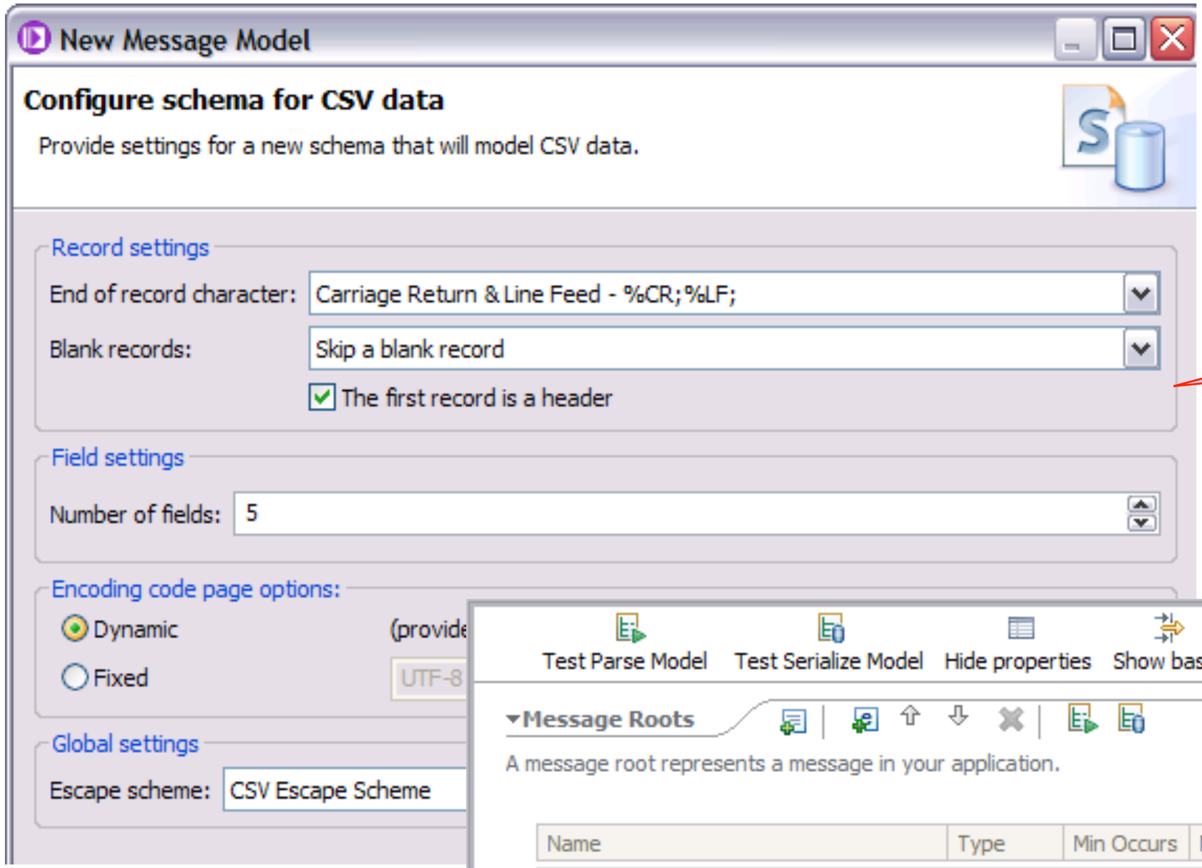
Comma Separated Values data is modeled by Data Format Description Language (DFDL) schema files. DFDL is a standard from the Open Grid Forum for describing all kinds of text and binary data.

- [✓] 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

```
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
```

Creating a DFDL model using guided authoring



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:	Skip a blank record
	✓ The first record is a header

Field settings

Number of fields:	5

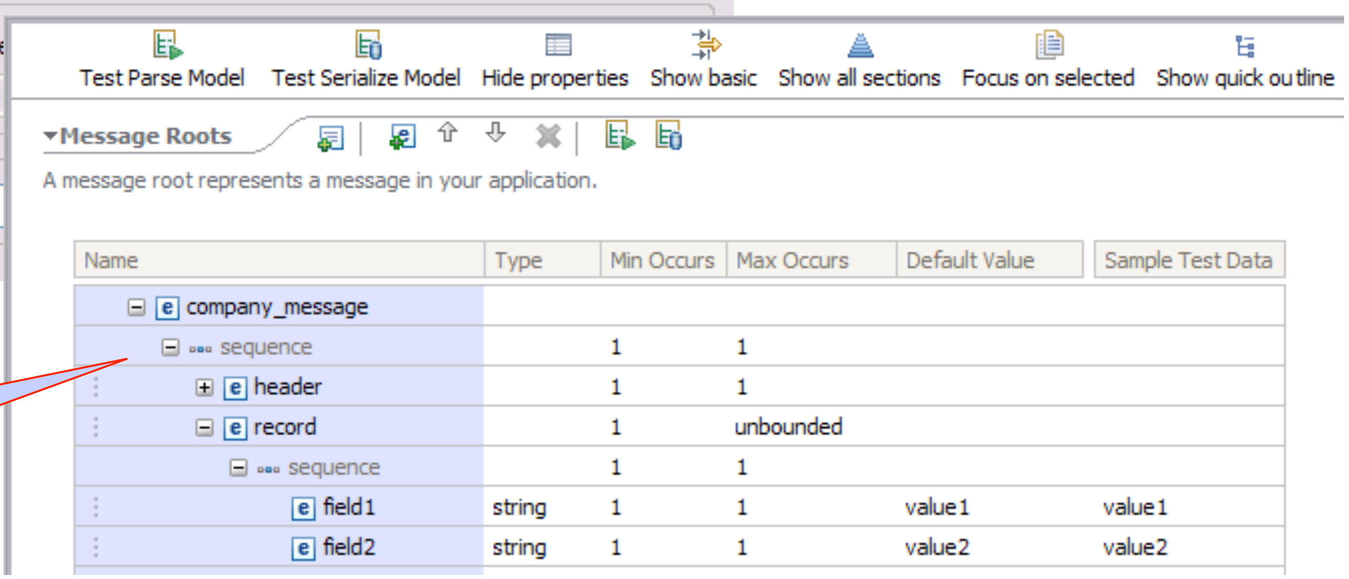
Encoding code page options:

[● Dynamic] (provide

○ Fixed UTF-8

Global settings

Escape scheme:	CSV Escape Scheme



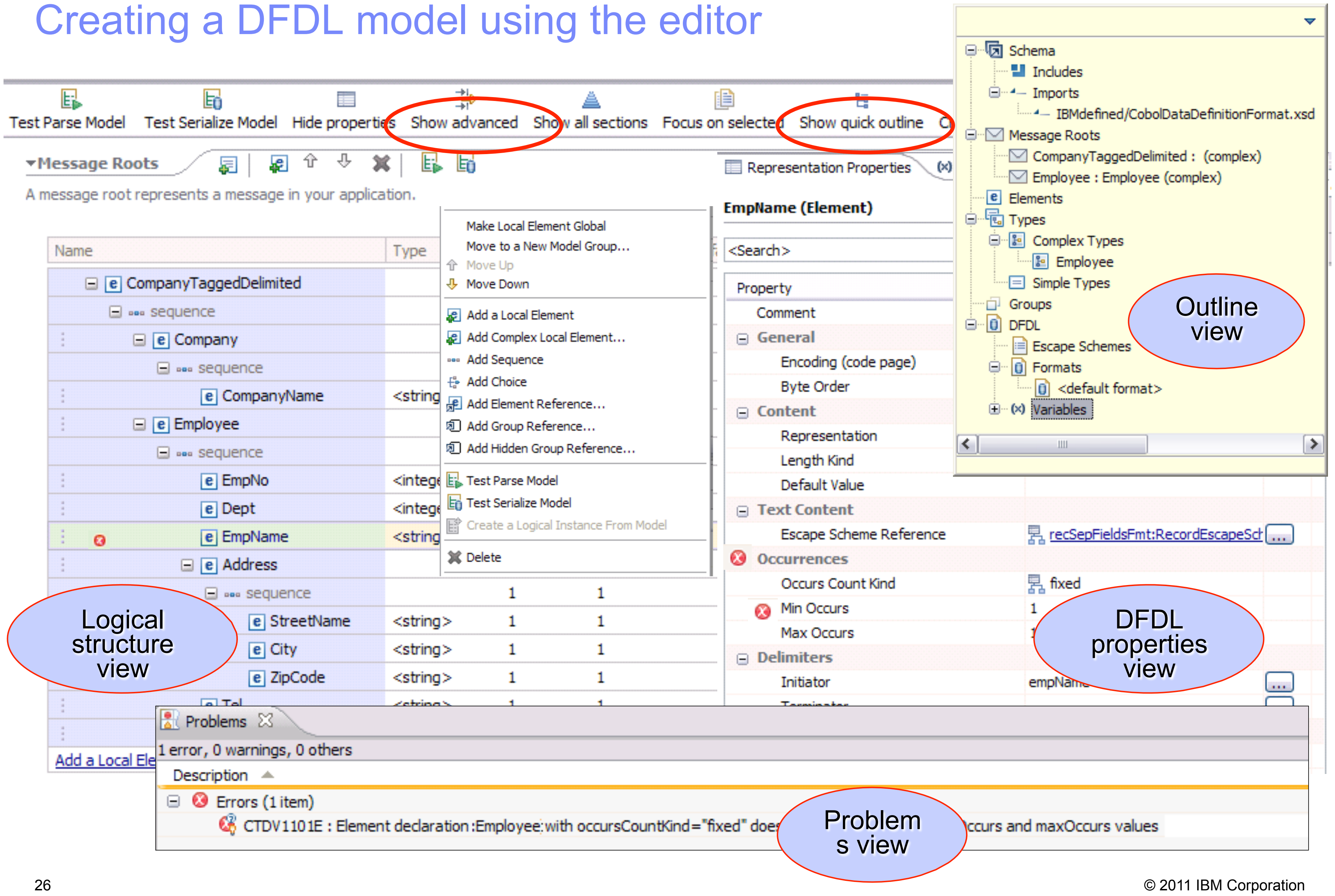
A message root represents a message in your application.

Name	Type	Min Occurs	Max Occurs	Default Value	Sample Test Data
company_message					
sequence		1	1		
header		1	1		
record		1	unbounded		
sequence		1	1		
field1	string	1	1	value1	value1
field2	string	1	1	value2	value2

Make selection

Generated model

Creating a DFDL model using the editor



Test Parse Model | Test Serialize Model | Hide properties | Show advanced | Show all sections | Focus on selected | Show quick outline

▾Message Roots

A message root represents a message in your application.

Name	Type		
CompanyTaggedDelimited			
sequence			
Company			
sequence			
CompanyName	<string		
Employee			
sequence			
EmpNo	<intege		
Dept	<intege		
EmpName	<string		
Address			
sequence		1	1
StreetName	<string>	1	1
City	<string>	1	1
ZipCode	<string>	1	1
Tel	<string>	1	1

Add a Local Ele

Make Local Element Global
Move to a New Model Group…
Move Up
Move Down
Add a Local Element
Add Complex Local Element…
Add Sequence
Add Choice
Add Element Reference…
Add Group Reference…
Add Hidden Group Reference…
Test Parse Model
Test Serialize Model
Create a Logical Instance From Model
Delete

Representation Properties

EmpName (Element)

<Search>

Property	
Comment	
General	
Encoding (code page)	
Byte Order	
Content	
Representation	
Length Kind	
Default Value	
Text Content	
Escape Scheme Reference	recSepFieldsFmt:RecordEscapeSch
Occurrences	
Occurs Count Kind	fixed
Min Occurs	1
Max Occurs	1
Delimiters	
Initiator	empName
Terminator	

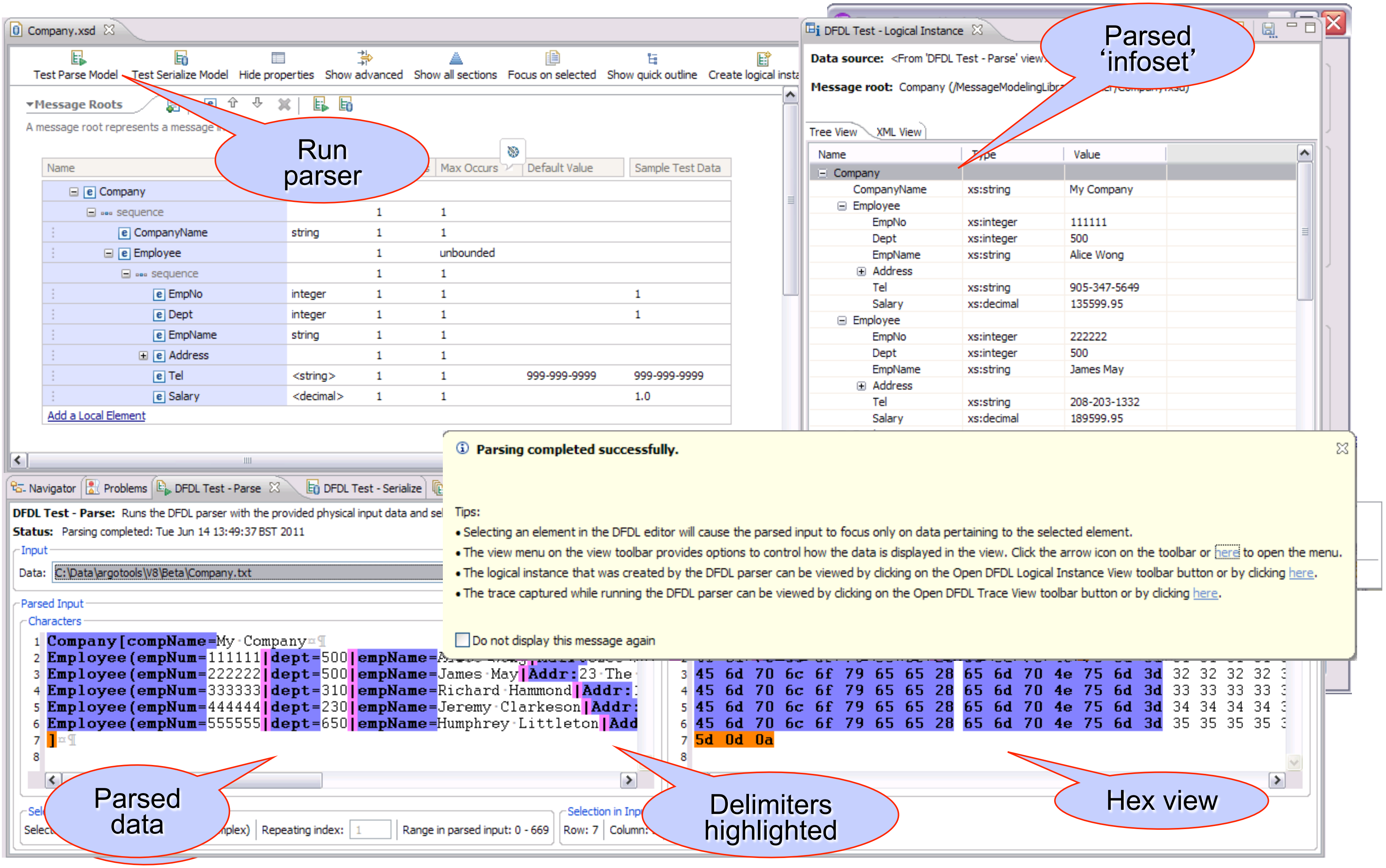
Schema
- Includes
- Imports
  - IBMdefined/CobolDataDefinitionFormat.xsd
- Message Roots
  - CompanyTaggedDelimited : (complex)
  - Employee : Employee (complex)
- Elements
- Types
  - Complex Types
    - Employee
  - Simple Types
- Groups
- DFDL
  - Escape Schemes
  - Formats
    - <default format>
  - Variables

Problems

1 error, 0 warnings, 0 others

Description
- Errors (1 item)
  - CTDV1101E : Element declaration :Employee with occursCountKind="fixed" does … ccurs and maxOccurs values

Testing a DFDL model within the editor



A message root represents a message

Name			Max Occurs	Default Value	Sample Test Data
Company					
sequence		1	1		
CompanyName	string	1	1		
Employee		1	unbounded		
sequence		1	1		
EmpNo	integer	1	1		1
Dept	integer	1	1		1
EmpName	string	1	1		
Address		1	1		
Tel	<string>	1	1	999-999-9999	999-999-9999
Salary	<decimal>	1	1		1.0

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

Message root: Company (/MessageModelingLibr

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		
EmpNo	xs:integer	222222
Dept	xs:integer	500
EmpName	xs:string	James May
Address		
Tel	xs:string	208-203-1332
Salary	xs:decimal	189599.95

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

Status: Parsing completed: Tue Jun 14 13:49:37 BST 2011

Input

Data: C:\Data\argotools\V8\Beta\Company.txt

Parsed Input

Characters

```
Company[compName=My·Company¤¶
Employee(empNum=111111|dept=500|empName=A
Employee(empNum=222222|dept=500|empName=James·May|Addr:23·The·
Employee(empNum=333333|dept=310|empName=Richard·Hammond|Addr:
Employee(empNum=444444|dept=230|empName=Jeremy·Clarkeson|Addr:
Employee(empNum=555555|dept=650|empName=Humphrey·Littleton|Add
]¤¶
```

```
3 45 6d 70 6c 6f 79 65 65 28 65 6d 70 4e 75 6d 3d 32 32 32 32
4 45 6d 70 6c 6f 79 65 65 28 65 6d 70 4e 75 6d 3d 33 33 33 33
5 45 6d 70 6c 6f 79 65 65 28 65 6d 70 4e 75 6d 3d 34 34 34 34
6 45 6d 70 6c 6f 79 65 65 28 65 6d 70 4e 75 6d 3d 35 35 35 35
7 5d 0d 0a
8
```

Sel Select plex) | Repeating index: 1 | Range in parsed input: 0 - 669

Selection in Inp Row: 7 | Column:

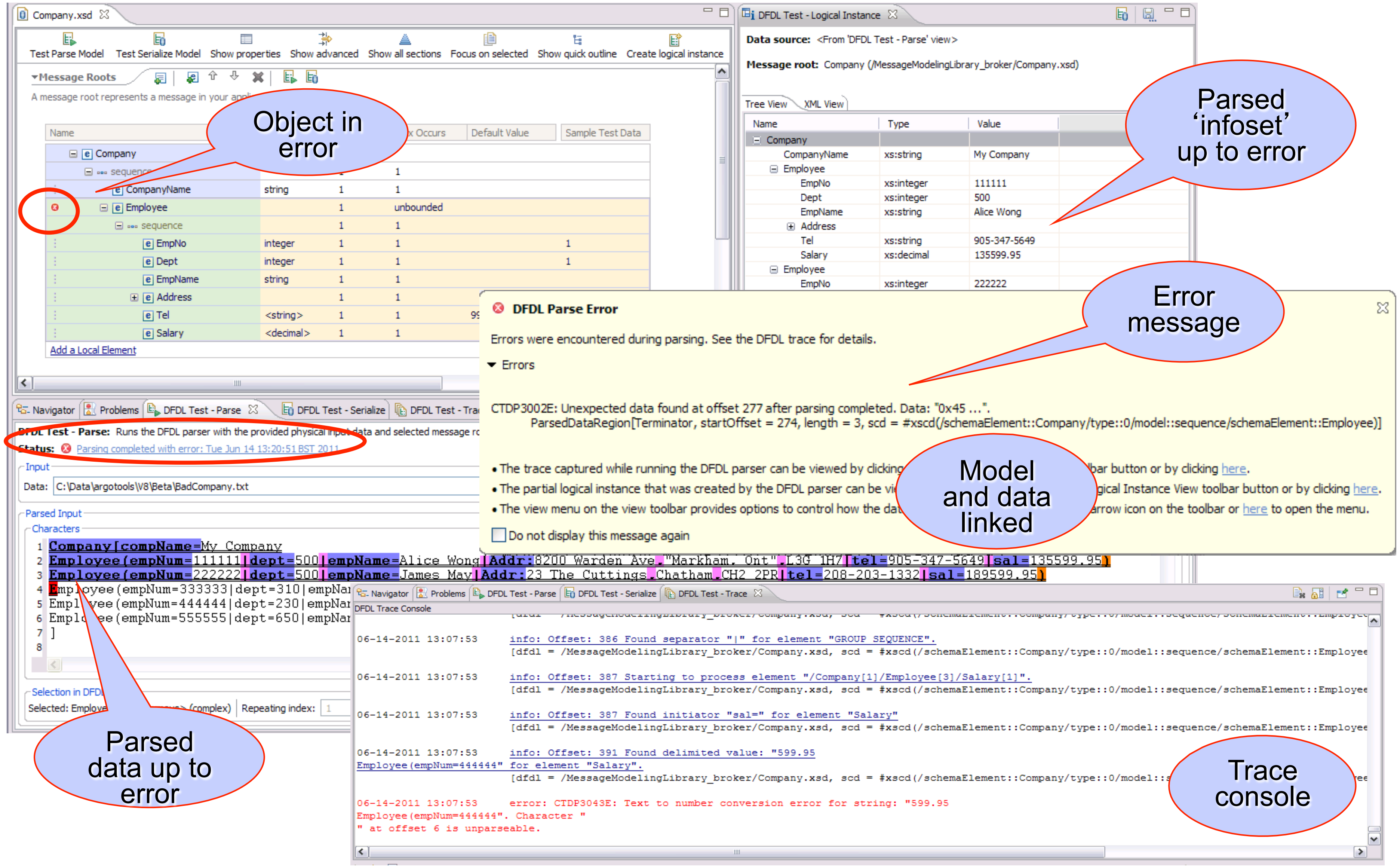
Parsing completed successfully.

Tips:

- Selecting an element in the DFDL editor will cause the parsed input to focus only on data pertaining to the selected 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.
- The logical instance that was created by the DFDL parser can be viewed by clicking on the Open DFDL Logical Instance View toolbar button or by clicking here.
- The trace captured while running the DFDL parser can be viewed by clicking on the Open DFDL Trace View toolbar button or by clicking here.

☐ Do not display this message again

Debugging a DFDL model test failure



A message root represents a message in your appl

Name			Max Occurs	Default Value	Sample Test Data
Company					
sequence			1		
CompanyName	string	1	1		
Employee		1	unbounded		
sequence		1	1		
EmpNo	integer	1	1		1
Dept	integer	1	1		1
EmpName	string	1	1		
Address		1	1		
Tel	<string>	1	1		99
Salary	<decimal>	1	1		

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

Message root: Company (/MessageModelingLibrary_broker/Company.xsd)

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		
EmpNo	xs:integer	222222

DFDL Parse Error

Errors were encountered during parsing. See the DFDL trace for details.

Errors

CTDP3002E: Unexpected data found at offset 277 after parsing completed. Data: "0x45 ...".
ParsedDataRegion[Terminator, startOffset = 274, length = 3, scd = #xscd(/schemaElement::Company/type::0/model::sequence/schemaElement::Employee)]

- The trace captured while running the DFDL parser can be viewed by clicking … bar button or by clicking here.
- The partial logical instance that was created by the DFDL parser can be vi… gical Instance View toolbar button or by clicking here.
- The view menu on the view toolbar provides options to control how the dat… arrow icon on the toolbar or here to open the menu.

☐ Do not display this message again

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

Status: Parsing completed with error: Tue Jun 14 13:20:51 BST 2011

Input

Data: C:\Data\argotools\V8\Beta\BadCompany.txt

Parsed Input

Characters

```
Company[compName=My Company
Employee(empNum=111111|dept=500|empName=Alice Wong|Addr:8200 Warden Ave,"Markham, Ont",L3G 1H7|tel=905-347-5649|sal=135599.95)
Employee(empNum=222222|dept=500|empName=James May|Addr:23 The Cuttings,Chatham,CH2 2PR|tel=208-203-1332|sal=189599.95)
Employee(empNum=333333|dept=310|empNa
Employee(empNum=444444|dept=230|empNa
Employee(empNum=555555|dept=650|empNa
]
```

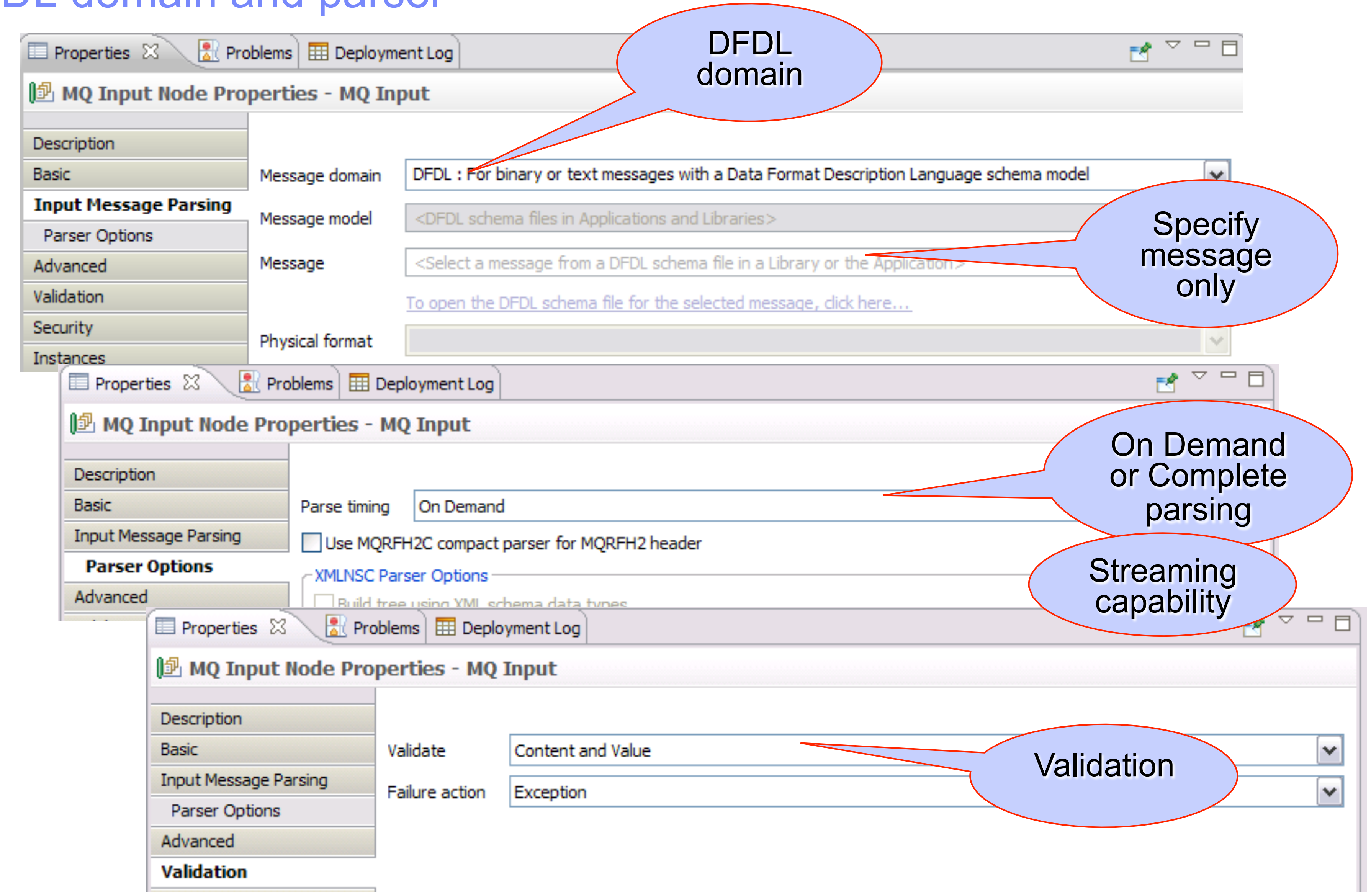
Selection in DFDL

Selected: Employee (complex) | Repeating index: 1

DFDL Trace Console

```
06-14-2011 13:07:53      info: Offset: 386 Found separator "|" for element "GROUP SEQUENCE".
                         [dfdl = /MessageModelingLibrary_broker/Company.xsd, scd = #xscd(/schemaElement::Company/type::0/model::sequence/schemaElement::Employee

06-14-2011 13:07:53      info: Offset: 387 Starting to process element "/Company[1]/Employee[3]/Salary[1]".
                         [dfdl = /MessageModelingLibrary_broker/Company.xsd, scd = #xscd(/schemaElement::Company/type::0/model::sequence/schemaElement::Employee

06-14-2011 13:07:53      info: Offset: 387 Found initiator "sal=" for element "Salary"
                         [dfdl = /MessageModelingLibrary_broker/Company.xsd, scd = #xscd(/schemaElement::Company/type::0/model::sequence/schemaElement::Employee

06-14-2011 13:07:53      info: Offset: 391 Found delimited value: "599.95
Employee(empNum=444444" for element "Salary".
                         [dfdl = /MessageModelingLibrary_broker/Company.xsd, scd = #xscd(/schemaElement::Company/type::0/model::s

06-14-2011 13:07:53      error: CTDP3043E: Text to number conversion error for string: "599.95
Employee(empNum=444444". Character "
" at offset 6 is unparseable.
```

DFDL domain and parser



Properties | Problems | Deployment Log

MQ Input Node Properties - MQ Input

Description | Basic | **Input Message Parsing** | Parser Options | Advanced | Validation | Security | Instances

Message domain	DFDL : For binary or text messages with a Data Format Description Language schema model
Message model	<DFDL schema files in Applications and Libraries>
Message	<Select a message from a DFDL schema file in a Library or the Application>
	To open the DFDL schema file for the selected message, click here...
Physical format	

Properties | Problems | Deployment Log

MQ Input Node Properties - MQ Input

Description | Basic | Input Message Parsing | **Parser Options** | Advanced

Parse timing	On Demand

☐ Use MQRFH2C compact parser for MQRFH2 header

XMLNSC Parser Options

☐ Build tree using XML schema data types

Properties | Problems | Deployment Log

MQ Input Node Properties - MQ Input

Description | Basic | Input Message Parsing | Parser Options | Advanced | **Validation**

Validate	Content and Value
Failure action	Exception

DFDL domain message tree

```
( ['MQROOT' : 0xd6d218]
  (0x01000000:Name):Properties = ( ['MQPROPERTYPARSER' : 0x141d34e8
    (0x03000000:NameValue):MessageSet              = `company.xsd' (CHARACTER)
    (0x03000000:NameValue):MessageType             = '{}:company_msg' (CHARACTER)
         000000:NameValue):MessageFormat           = '' (CHARACTER)
           0000:NameValue):Encoding                = 273 (INTEGER)
          00000:NameValue):CodedCharSetId          = 850 (INTEGER)
    ....
  )
  (0x01000000:Name):DFDL       = ( ['dfdl' : 0xd812
    (0x01000000:Name):company_msg = (
      (0x01000000:Name):header = (
        (0x03000000:NameValue):company  = 'IBM' (CHARACTE
      )
      (0x01000000:Name):employee = (
        (0x03000000:NameValue):name     = 'Steve Hanson' (CHARACTER)
        (0x03000000:NameValue):age      = 47 (INTEGER)
        (0x03000000:NameValue):dept     = 12345 (INTEGER)
        (0x03000000:NameValue):contract = FALSE (BOOLEAN)
        (0x03000000:NameValue):start    = '1988-10-30 09:00:00' (TIMESTAMP)
      )
    )
  )
)
```

DFDL schema and message name

DFDL domain

Message name in tree (like XMLNSC)

Compact ‘Name/Value’ syntax elements

Data types from DFDL schema

Setting defaults for DFDL properties

- In the DFDL language, DFDL properties do ***not*** have built-in defaults
- This is a deliberate design decision to avoid behavioural differences when switching platforms and locales
- If an object needs a property, a value must be explicitly supplied for the property

1. You can set the property locally on the object itself
2. You can set the property on a dfdl:format annotation within a named, shareable dfdl:defineFormat annotation, and reference the dfdl:defineFormat using the special dfdl:ref property
3. You can set the property in the schema’s special dfdl:format annotation, where it acts as a default for all objects in the schema

- The dfdl:ref property can also be used on dfdl:format, enabling inheritance chaining
- In DFDL, using dfdl:format to set property defaults in this way is called ***scoping***

An example of DFDL scoping

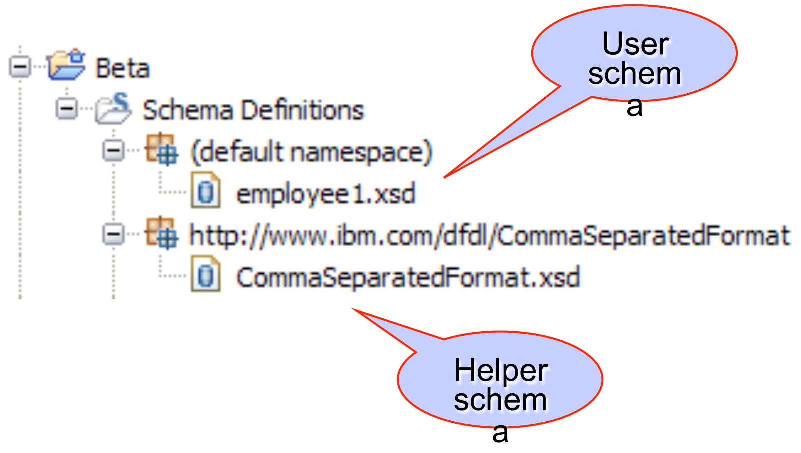
```
                 a:xxx,b:yyy,c:zzz
<xs:schema>
  <xs:annotation>
   <xs:appinfo source="http://www.ogf.org/dfdl/" >
      <dfdl:format initiator="c:" … />
   </xs:appinfo>
  </xs:annotation>

  <xs:annotation>
   <xs:appinfo source="http://www.ogf.org/dfdl/" >
      <dfdl:defineFormat name="myFormat" />
         <dfdl:format initiator="b:" … />
      </dfdl:defineFormat>
   </xs:appinfo>
  </xs:annotation>

  <xs:complexType>
    <xs:sequence dfdl:separator="," dfdl:initiator="">
      <xs:element name="a" type="xs:string" dfdl:initiator="a:" />
      <xs:element name="b" type="xs:string"  dfdl:ref="myFormat" />
      <xs:element name="c" type="xs:string" />
    </xs:sequence>
   </xs:complexType>
</xs:schema>
```

Property set locally

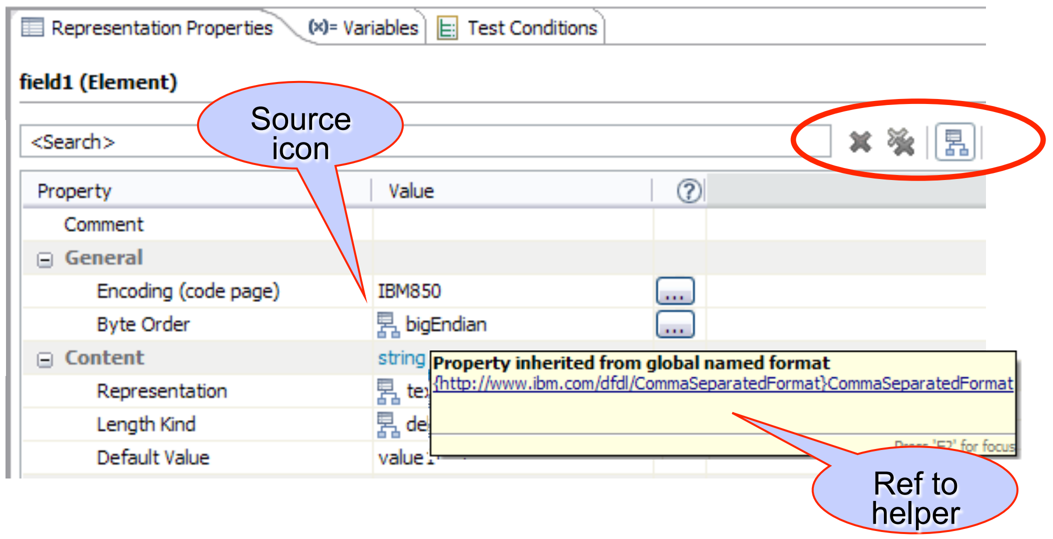
Property via dfdl:ref

Property from schema's dfdl:format

Wizards help with scoping



- Beta
  - Schema Definitions
    - (default namespace)
      - employee1.xsd
    - http://www.ibm.com/dfdl/CommaSeparatedFormat
      - CommaSeparatedFormat.xsd



Representation Properties | Variables | Test Conditions

field1 (Element)

<Search>

Property	Value
Comment	
General	
Encoding (code page)	IBM850
Byte Order	bigEndian
Content	string
Representation	te
Length Kind	de
Default Value	value

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

Press 'F2' for focus

- As we have learned, DFDL properties do ***not*** have built-in defaults, if an object needs a property, a value must be supplied
- To ease this task, the wizard automatically creates a helper DFDL schema that contains a named dfdl:defineFormat with values set for most DFDL properties, and adds it as an import into the user DFDL schema
- The user DFDL schema has a dfdl:format that references the dfdl:defineFormat in the helper schema using dfdl:ref, thereby setting up defaults for all objects in the user schema
- There is a different helper DFDL schema for each kind of data (eg, COBOL, CSV, …)
- Helper DFDL schemas are created only once per Message Broker library
- Helper DFDL schemas are created read-only (although they can be edited using the DFDL editor in the usual way if you need to)

DFDL features supported

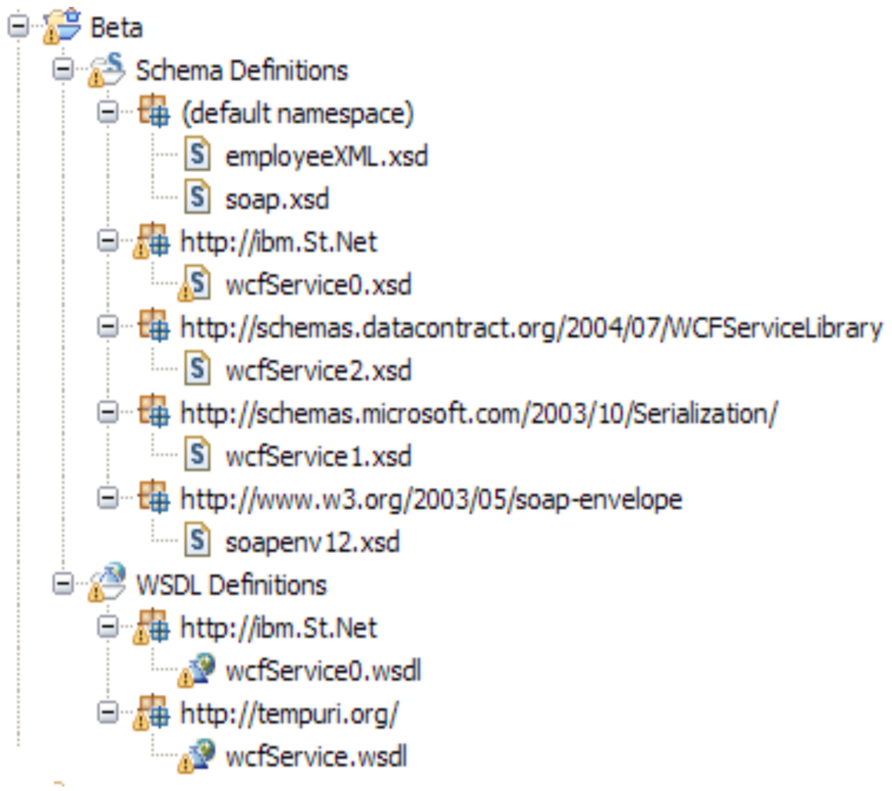
- DFDL support in MB implements the OGF DFDL 1.0 specification
- The vast majority of DFDL features are supported at MB V8 FP1
- Some more advanced features of DFDL are not yet available and will be added in future MB deliverables
- Currently unsupported:
  - Bi-directional text
  - Unordered groups & floating elements
  - Occurs Count Kind ‘parsed’ and ‘stopValue’
  - Length Kind ‘pattern’ and ‘endOfParent’
  - Asserts & Discriminators with testKind ‘pattern’
  - Separator Policy ‘suppressedAtEndStrict’
  - Certain XPath functions
  - Delimited binary data
  - Arrays with stop values
  - Default values (parser)
  - User-defined variables
  - Hidden groups
  - Calculated values

Agenda

- DFDL – a new standard for modeling text and binary data
- DFDL support in Message Broker
- Other enhancements
- Message Sets and MRM

Other message modeling enhancements

- XML Schema and WSDL files can reside in a Message Broker library
  - For XMLNSC, SOAP, DataObject domains this is recommended instead of using a Message Set project
  - Can drop XSD and WSDL files straight into a Message Broker library
  - Any additional SOAP schemas are automatically added
  - Use built-in XML Schema and WSDL editors for editing files
- New Message Model wizard also used for creating models for all kinds of data
  - XML
  - Enterprise Information Systems
  - SCA
  - CORBA
  - MIME
  - Database



```
Beta
  Schema Definitions
    (default namespace)
      employeeXML.xsd
      soap.xsd
    http://ibm.St.Net
      wcfService0.xsd
    http://schemas.datacontract.org/2004/07/WCFServiceLibrary
      wcfService2.xsd
    http://schemas.microsoft.com/2003/10/Serialization/
      wcfService1.xsd
    http://www.w3.org/2003/05/soap-envelope
      soapenv12.xsd
  WSDL Definitions
    http://ibm.St.Net
      wcfService0.wsdl
    http://tempuri.org/
      wcfService.wsdl
```

Agenda

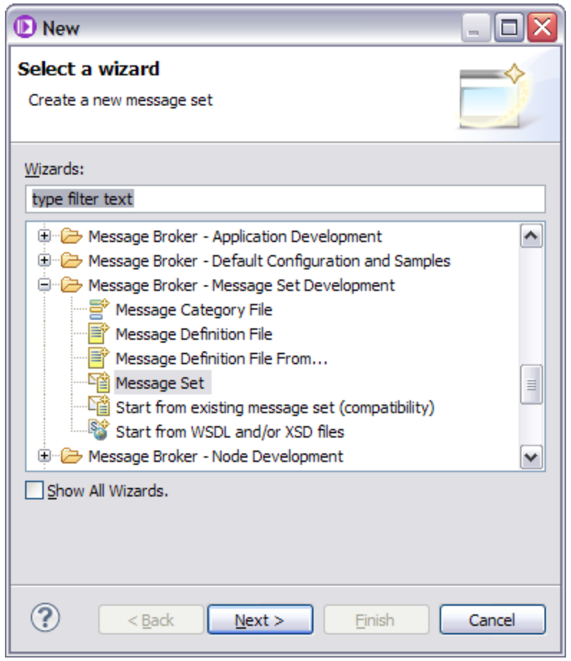
- DFDL – a new standard for modeling text and binary data
- DFDL support in Message Broker
- Other enhancements
- Message Sets and MRM

Message Sets and MRM

- Message Set projects continue to be supported
  - For all existing domains
  - Required for MRM & IDOC domains
- Message Set projects must be first created as an Independent Resource
  - Can then be included in an Application or a Library
  - Or can remain independent
- MXSD files can not reside directly in a Message Broker project or Library
- DFDL schema files can not reside in a Message Set project
- MRM domain continues to be supported
  - Can be used with new Graphical Data Mapper
- Currently no auto-migration of MRM physical formats to DFDL

Message Set wizards

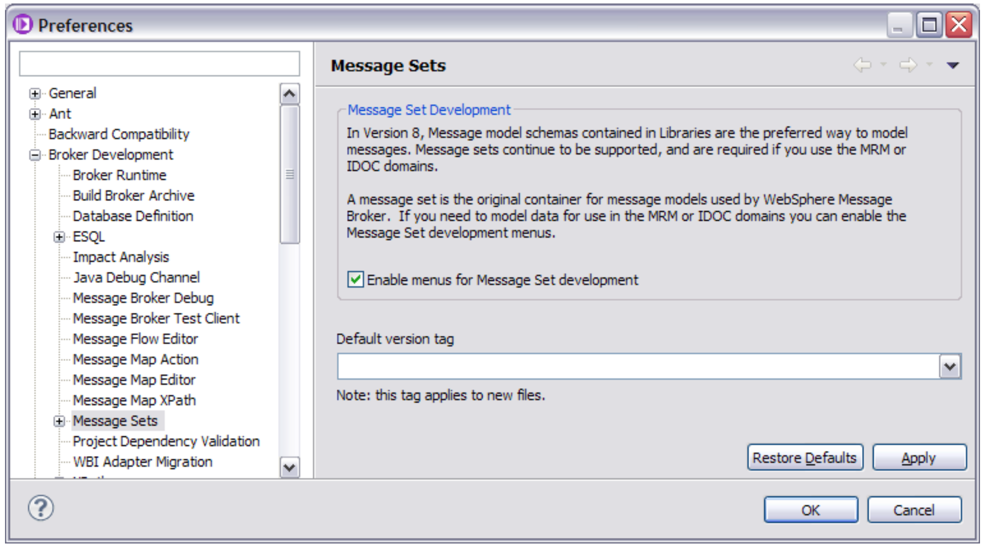
- New message set wizard entry points de-emphasized
  - Available by default via New->Other
- Message Set Preference to increase visibility
  - ✓ Enable menus for Message Set development
- Adds message set entry points to context menus
  - New->Independent Resources->Message Set
  - New->Message Definition File from a message set


New

Select a wizard

Create a new message set

Wizards:
type filter text
Message Broker - Application Development
Message Broker - Default Configuration and Samples
Message Broker - Message Set Development
Message Category File
Message Definition File
Message Definition File From...
Message Set
Start from existing message set (compatibility)
Start from WSDL and/or XSD files
Message Broker - Node Development
Show All Wizards.
< Back
Next >
Finish
Cancel


Preferences

Message Sets

Message Set Development

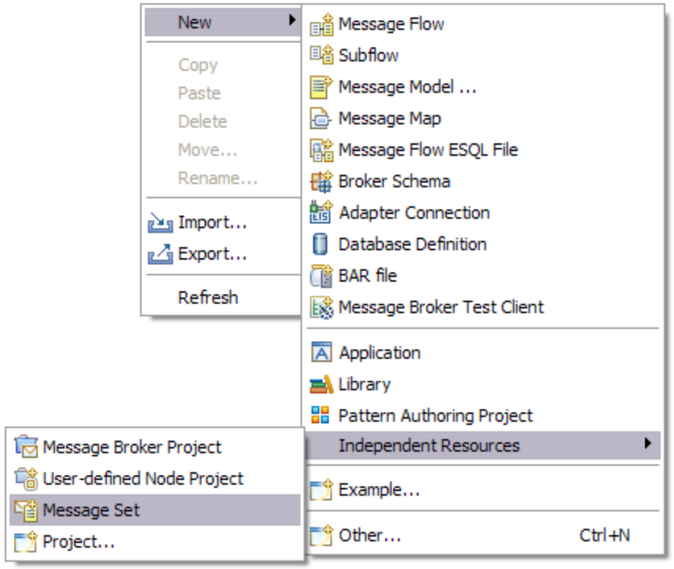
In Version 8, Message model schemas contained in Libraries are the preferred way to model messages. Message sets continue to be supported, and are required if you use the MRM or IDOC domains.

A message set is the original container for message models used by WebSphere Message Broker. If you need to model data for use in the MRM or IDOC domains you can enable the Message Set development menus.

[✓] Enable menus for Message Set development

Default version tag

Note: this tag applies to new files.



धन्यवाद
Hindi

多謝
Traditional Chinese

Grazie
Italian

ขอบคุณ
Thai

Спасибо
Russian

Obrigado
Brazilian Portuguese

Merci
French

Gracias
Spanish

شكراً
Arabic

多谢
Simplified Chinese

Danke
German

நன்றி
Tamil

ありがとうございました
Japanese

감사합니다