IBM WebSphere Message Broker Modelling and Parsing Business Data

Tim Kimber,
WebSphere Message Broker Development
IBM Hursley Park, UK
Agenda

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

- DFDL – a new standard for modeling text and binary data
- DFDL support in Message Broker
- Message Sets and MRM
Data modeling – where we are today

- No universal standard for modeling general text and binary data
  - XML -> use XML Schema
  - RDBMS -> use database schema
  - Text/binary -> ??

- IBM products have their own way of modeling text and binary data which has evolved over time 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

- No common set of industry models

- A new open modeling standard called DFDL aims to change this!
Data Format Description Language (DFDL)

- A new open standard
  - From the Open Grid Forum (OGF)
  - http://www.ogf.org/
  - Version 1.0
  - ‘Proposed Recommendation’ status

- A way of describing data…
  - It is NOT a data format itself!

- That can describe any data format …
  - Textual and binary
  - Commercial record-oriented
  - Scientific and numeric
  - Modern and legacy
  - Industry standards

- While allowing high performance …
  - Choose the right data format for the job

- Leverage XML technology and concepts
  - Use W3C XML Schema subset & type system to describe the logical format of the data
  - Use annotations within the XSD to describe the physical representation of the data
  - Use XPath when referencing fields within the data

- Support round-tripping
  - Read and write data in described format from same description

- Keep simple cases simple
  - Simple descriptions should be human readable

- Generality
  - Think “Type Tree + MRM” & more
Example – Delimited text data

Separators, initiators (aka tags), & terminators are all examples in DFDL of delimiters.
Example – DFDL schema

```xml
<xsd:complexType name="myNumbers">
  <xsd:sequence>
    <xsd:annotation>
      <xsd:appinfo source="http://www.ogf.org/dfdl/v1.0">
        <dfdl:sequence separator=";" encoding="ascii"/>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:element name="myInt" type="xsd:int">
      <xsd:annotation>
        <xsd:appinfo source="http://www.ogf.org/dfdl/v1.0">
          <dfdl:element representation="text" textNumberRep="standard" encoding="ascii" lengthKind="delimited" initiator="int=" .../>
        </xsd:appinfo>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="myFloat" type="xsd:float">
      <xsd:annotation>
        <xsd:appinfo source="http://www.ogf.org/dfdl/v1.0">
          <dfdl:element representation="text" textNumberRep="standard" encoding="ascii" lengthKind="delimited" initiator="float=" .../>
        </xsd:appinfo>
      </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```
Example – DFDL schema (short form)

```xml
<xs:complexType name="myNumbers">
  <xs:sequence dfdl:separator=";" dfdl:encoding="ascii">
    <xs:element name="myInt" type="xs:int"
      dfdl:representation="text"
      dfdl:textNumberRep="standard" dfdl:encoding="ascii"
      dfdl:lengthKind="delimited" dfdl:initiator="int=" ... />
    <xs:element name="myFloat" type="xs:float"
      dfdl:representation="text"
      dfdl:textNumberRep="standard" dfdl:encoding="ascii"
      dfdl:lengthKind="delimited" dfdl:initiator="float=" ... />
  </xs:sequence>
</xs:complexType>
```

DFDL properties
DFDL features

- Language structures such as COBOL, C and PL/1
- Industry standards such as SWIFT, HL7, FIX, HIPAA, X12, EDIFACT, ISO8583
- Fixed data and data delimited by text or binary markup
- Text data types such as strings, numbers, zoned decimals, calendars, booleans
- Binary data types such as integers, floats, BCD, packed decimal, calendars, booleans
- 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
- XPath 2.0 expression language including variables to model dynamic data
- Speculative parsing to resolve choices and optional content
- Fixed and variable arrays
- Hide elements in the data
- Calculate element values
- Validation to XML Schema 1.0 rules
- Scoping mechanism to allow common property values to be applied at multiple points
DFDL language - objects

DFDL properties are placed on yellow objects only
## DFDL language – basic annotations

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Used on Component</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfdl:element</td>
<td>xs:element</td>
<td>Contains the DFDL properties of an xs:element and xs:element reference</td>
</tr>
<tr>
<td></td>
<td>xs:element reference</td>
<td></td>
</tr>
<tr>
<td>dfdl:choice</td>
<td>xs:choice</td>
<td>Contains the DFDL properties of an xs:choice.</td>
</tr>
<tr>
<td>dfdl:sequence</td>
<td>xs:sequence</td>
<td>Contains the DFDL properties of an xs:sequence.</td>
</tr>
<tr>
<td>dfdl:group</td>
<td>xs:group reference</td>
<td>Contains the DFDL properties of an xs:group reference to a group definition containing an xs:sequence or xs:choice.</td>
</tr>
<tr>
<td>dfdl:simpleType</td>
<td>xs:simpleType</td>
<td>Contains the DFDL properties of an xs:simpleType</td>
</tr>
<tr>
<td>dfdl:format</td>
<td>xs:schema</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>dfdl:defineFormat</td>
<td></td>
</tr>
<tr>
<td>dfdl:defineFormat</td>
<td>xs:schema</td>
<td>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.</td>
</tr>
</tbody>
</table>
# DFDL language – more advanced annotations

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


Agenda

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

Message Sets and MRM
DFDL support in runtime and toolkit

- DFDL models
  - Schema files reside in Message Broker libraries
  - NOT in Message Set projects

- DFDL domain and model-driven parser
  - Available in all nodes, just like other MB domains and parsers
  - Use instead of MRM CWF/TDS
    - More capable and higher performing than MRM

- Tooling for creating DFDL models
  - Guided authoring wizards
  - Language importer wizards
  - DFDL editor

- DFDL model debugger
  - Debug parsing and writing of data from within the MB toolkit
  - No deploy necessary!

- DFDL models deployed to broker in BAR file
  - No dictionary file!
Creating a DFDL model

- 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
Creating a DFDL model using guided authoring

**Generated model**

- **Make selection**

- **New Message Model**
  - **Configure schema for 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
    - **Global settings**
      - Escape scheme: CSV Escape Scheme

- **Message Roots**
  - A message root represents a message in your application.
  - | Name             | Type  | Min Occurs | Max Occurs | Default Value | Sample Test Data |
  - |------------------|-------|------------|------------|---------------|-----------------|
  - | company_message  |       | 1          | 1          |               |                 |
  - | sequence         |       | 1          | 1          |               |                 |
  - | header           |       | 1          | 1          |               |                 |
  - | record           |       | 1          | unbounded  |               |                 |
  - | sequence         |       | 1          | 1          |               |                 |
  - | field1           | string| 1          | 1          | value1        | value1          |
  - | field2           | string| 1          | 1          | value2        | value2          |
Creating a DFDL model using the editor

Outline view

Logical structure view

DFDL properties view

Problem s view
Testing a DFDL model within the editor

Run parser

Parsed 'infoset'

Parsed data

Delimiters highlighted

Hex view
Debugging a DFDL model test failure

Object in error

Parsed 'infoset' up to error

Error message

Model and data linked

Parsed data up to error

Trace console
DFDL domain and parser

Specify schema & message

On Demand or Complete parsing

Streaming capability

Validation
The DFDL parser is smart!

- The DFDL parser is a recursive-descent parser with look-ahead used to resolve points of uncertainty:
  - A choice
  - An optional element
  - A variable array of elements

- Put another way, the DFDL parser speculatively attempts to parse data until an object is either ‘known to exist’ or ‘known not to exist’

- Until that applies, the occurrence of a processing error causes the parser to suppress the error, back track and make another attempt

- Example: The parser tries the 1st branch of a choice, but gets a processing error. It back tracks and tries the 2nd branch, which succeeds

- The dfdl:discriminator annotation can be used to assert that an object is ‘known to exist’, which prevents incorrect back tracking

- The presence of an initiator (tag) is also able to assert ‘known to exist’
DFDL message tree

( ['MQROOT' : 0xd6d218]
  (0x01000000:Name):Properties = ( ['MQPROPERTYPARSER' : 0x141d34e8]
    (0x03000000:NameValue):MessageSet = 'company.xsd' (CHARACTER)
    (0x03000000:NameValue):MessageType = '{}' (CHARACTER)
    (0x03000000:NameValue):MessageFormat = '' (CHARACTER)
    (0x03000000:NameValue):Encoding = 273 (INTEGER)
    (0x03000000:NameValue):CodedCharSetId = 850 (INTEGER)
  )
  (0x01000000:Name):company_msg = ( ['dfdl' : 0xd812c8]
    (0x01000000:Name):header = ( (0x03000000:NameValue):company = 'IBM' (CHARACTER)
    )
    (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)
    )
  )
)
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
```

```xml
<x: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:" ... />
    </xs:defineFormat>
  </xs:appinfo>

  <xs:complexType>
    <xs:sequence dfdl:separator=""," >
      <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>
```
Wizards help with scoping

- 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 (e.g., 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 GA
- Some more advanced features of DFDL are not yet available, these are listed in the MB infocenter
- Support for these will be added in a future MB deliverable
Agenda

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

- Message Set projects continue to be supported
  - For all existing domains
  - Required for MRM domain

- New Message Set and New Message Definition File wizard context menus are enabled via a Preference setting
  - By default they are only accessible using New->Other->Message Broker -> Message Set Development

- MXSD files cannot reside in a Message Broker project or library

- DFDL schema files cannot reside in, or be imported into, a Message Set

- MRM domain continues to be supported
  - Can be used with new Graphical Data Mapper

- No automatic migration to DFDL planned
Any questions?
IBM, the IBM logo, ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.