DFDL WG Session 1 Summary of Status

Mike Beckerle

Ascential Software

DFDL-WG Session 1
Summary of Status

Agenda

- Presentation: primer material (45 mins)
  - Review of purposes/goals
  - XML / XSD impact: Data Model
  - Examples from primer
- General discussion (40 mins)
- Overview of other sessions (5 mins)

Data Interchange Formats

- Prescriptive: *Put your data in this format!*
  - XML – textual
  - Binary – ASN.1, XDR, NetCDF, HDF...

- Descriptive: *What format is your data in?*
  - Various commercial products. Nothing standard.
  - **DFDL**

Why Descriptive?

Allows us to achieve two goals simultaneously:

1. Interoperability
   - Modern and Legacy data formats
2. *Performance!*
   - Density
     - Fewest bytes to represent data without resorting to compression
   - Optimized I/O
     - Seekable random access
     - Memory mapped, aligned
       - Without sacrificing general access

Why the GGF for DFDL?

- Grids are about big-data and big-computation problems
  - Simplistic solutions like “use XML” won’t cut it!
- Grids are about universal data interchange

Related Standards Efforts

- Prescriptive systems:
  - W3C binary XML (http://www.w3.org/XML/Binary/)
    - Formed, but discussion group has no items.
- Descriptive systems:
    - None known

XML Synergy

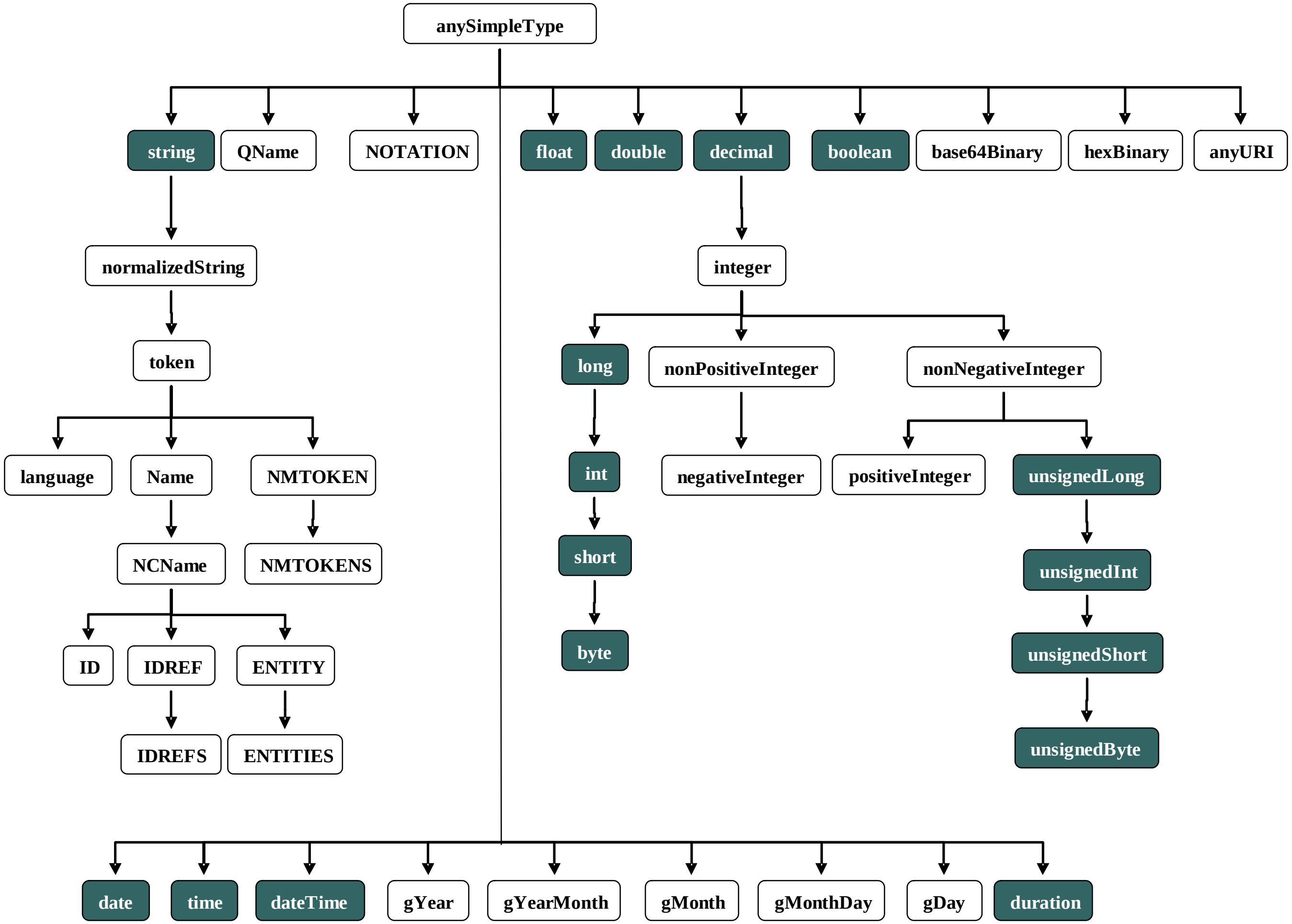
¢ Use XSD to describe the logical data

¢ Use annotations within the XSD to describe the representation of it.

XSD Types

- Elements
  - A.k.a. fields
- Sequence groups, All groups
  - All = unordered group
- Choice
  - A.k.a. union, redefine,
- Vectors
  - Use element with minOccurs, maxOccurs.
- Nillability
  - A.k.a. Nullable values

XML/XSD – basic types



Example 1: XML

```
<w>5</w>
<x>7839372</x>
<y>8.6E-200</y>
<z>-7.1E8</z>
```

Example 1: XSD

```
<xs:sequence>
        <xs:element name="w" type=“int"/>
        <xs:element name="x" type=“int"/>
        <xs:element name="y" type=“double"/>
        <xs:element name="z" type=“float"/>
</xs:sequence>
```

Example 1 DFDL - binary

```
0000 0005 0077 9e8c
169a 54dd 0a1b 4a3f
ce29 46f6
```

Example 1 DFDL - binary

```
    <xs:complexType name="example1">
    <xs:annotation>
        <xs:appinfo>
            <binaryProperties>
                <byteOrder>bigEndian</byteOrder>
            </binaryProperties>
        </xs:appinfo>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="w" type="dfdl:binaryInt"/>
        <xs:element name="x" type="dfdl:binaryInt"/>
        <xs:element name="y" type="dfdl:binaryDouble"/>
        <xs:element name="z" type="dfdl:binaryFloat"/>
    </xs:sequence>
</xs:complexType>
```

Example 1 DFDL - textual

```
“5, 7839372, 8.6E-200, -7.1E8”
```

Example 1 DFDL - textual

```
<xs:complexType name="example1">
    <xs:annotation>
        <xs:appinfo>
            <characterProperties>
                <characterSet>UTF-8</characterSet>
            </characterProperties>
            <numericTextProperties>
                <decimalSeparator>.</decimalSeparator>
            </numericTextProperties>
            <groupProperties>
                <fieldSeparator>,</fieldSeparator>
            </groupProperties>
        </xs:appinfo>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="w" type="dfdl:textInt"/>
        <xs:element name="x" type="dfdl:textInt"/>
        <xs:element name="y" type="dfdl:textDouble"/>
        <xs:element name="z" type="dfdl:textFloat"/>
    </xs:sequence>
</xs:complexType>
```

The pieces of the puzzle

- Primitive types: XSD
- Data structure: XSD
- Mappings – representations to/from primitives: DFDL mappings
- Composition – modular composition of basic mappings: DFDL mapping composition
- References – associate primitives in a structure with correct mapping: DFDL mapped types
- N.B. Problem parts have some level independence

Mappings

- Named black boxes
- Implementations know how to call them
- Semantics not described
  - Name
  - Range type
  - Domain type
  - Directionality

```
<definitions>
   <mapping name="dfdl:data-bytes"
             rangeType="dfdl:data"
             domainType="dfdl:bytes.unbounded"
             direction="bidirectional"/>
   <mapping name="dfdl:bytes-int"
             rangeType="dfdl:bytes.4"
             domainType="xs:int"
             direction="bidirectional"
             argumentType="dfdl:binaryProperties"/>
</definitions>
```

Composing mappings

- Simple linear composition (from primer)

```
<compositeMapping>
   <mapping name="databytes"/>
   <mapping name="bytes-int"/>
</compositeMapping>
```

- May be too restricted more complex composition (mapping trees) possible.

Mapped types

- Associate a new type with a composed mapping
- Use a trick, null restriction with annotation
- New type is a valid XML type
- Simple composition could be included in annotation

```
<xs:simpleType name="binaryInt">
   <xs:restriction base="xs:int">
       <xs:annotation>
           <xs:appinfo>
               <compositeMapping>
                   <mapping name="data-bytes"/>
                   <mapping name="bytes-int"/>
               </compositeMapping>
           </xs:appinfo>
       </xs:annotation>
   </xs:restriction>
</xs:simpleType>
```

About USE

- For convenience instead of textInt or binaryInt, you create a “use” association
- Means: All uses of type “xs:int” in the scope of this declaration mean “dfdl:binaryInt”

```
<use type="dfdl:binaryInt"/>
```

Example 1 – binary with ‘use’

```
    <xs:complexType name="example1">
    <xs:annotation>
        <xs:appinfo>
            <use type="dfdl:binaryInt"/>
            <use type="dfdl:binaryFloat"/>
            <use type="dfdl:binaryDouble"/>
        </xs:appinfo>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="w" type="int"/>
        <xs:element name="x" type="int"/>
        <xs:element name="y" type="double"/>
        <xs:element name="z" type="float"/>
    </xs:sequence>
</xs:complexType>
```

Next Steps

¢ Tuesday session

Open Issues

l Richer data formats: stored length, choice/unions

l Layered translation, modularity

¢ Wednesday session

Continuation on Open Issues

XML/XSD concerns