

Extra Slides

Unparsing: *dfd1:outputValueCalc*

- Property - defines an evaluation at unparse time that provides the value of an element before it is written out.
- Expressions can refer to parts of the Infoset that are after it.

Things DFDL (v1.0 + *BLOB*) Does

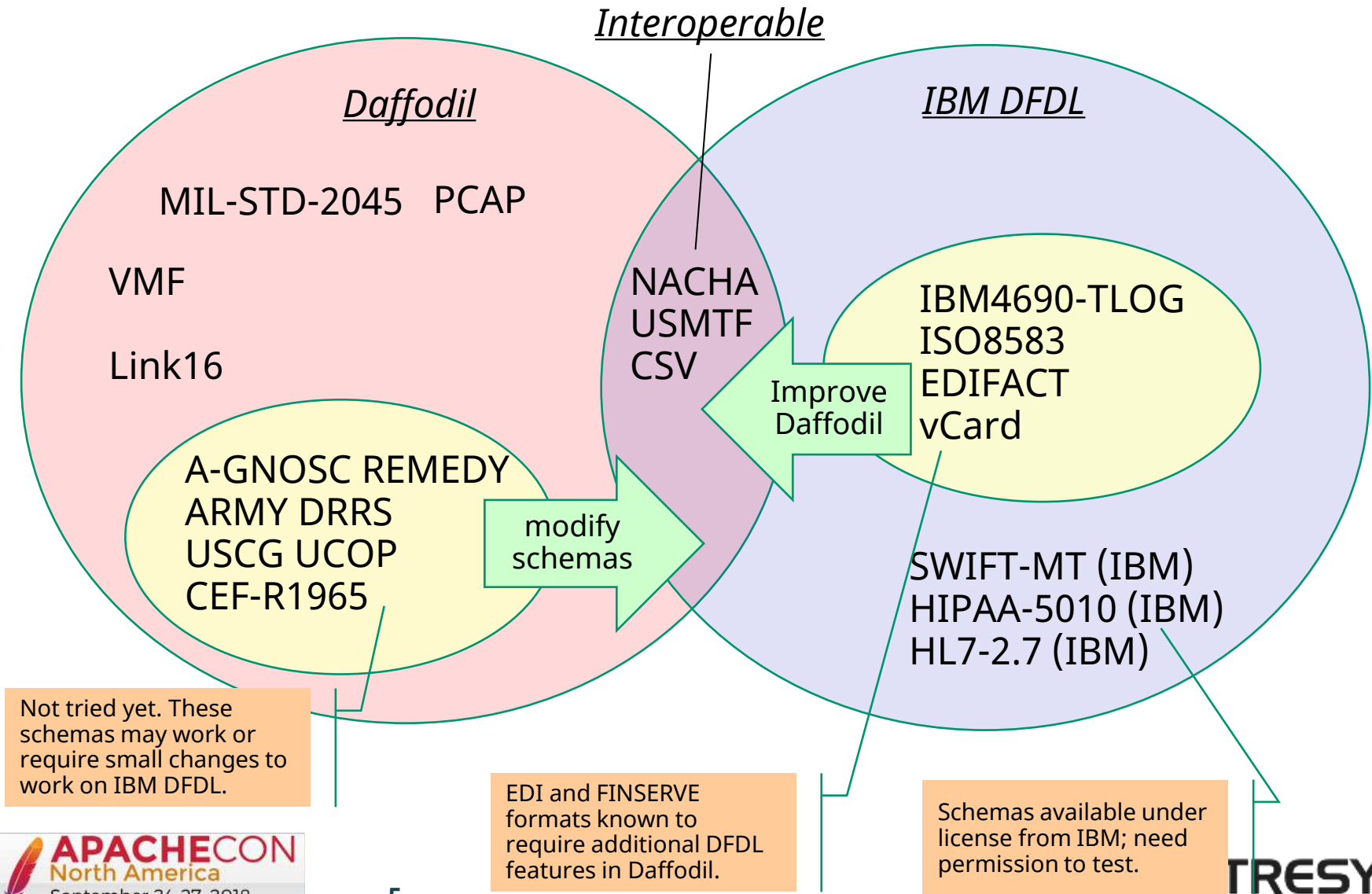
DFDL is for Images and Video

- Originally not in scope
- Large user demand to use DFDL on the metadata content of image file formats
 - Cybersecurity applications
- Adding BLOB (Binary Large Object) feature to DFDL language to enable DFDL to describe image files

Daffodil Future Development

- Cross-validation/test with IBM DFDL
 - Interop test on all IBM-created DFDL schemas
- Tutorials on writing/debugging DFDL schemas
- Improved trace and debug
- Full SAX-style streaming behavior for parsing, unparsing
- Faster schema compilation for large schemas
- Integrate into more frameworks
- Extensions: recursion, BLOB/CLOB, table-lookup,...

Interoperability – Daffodil & IBM



What I said I would talk about....

- DFDL - what it is,
- the standard,
- why it is important,
- what it does (and doesn't) do for systems that intake/export data,
- and how it is different from other approaches to this problem in the past.
- All of this motivates why we're incubating Daffodil.

The talk also covers technical status of Daffodil code base to give prospective contributors a bit of insight into what the code-base/effort is like.

Things DFDL (v1.0) Doesn't Do

That's not stopping anybody!

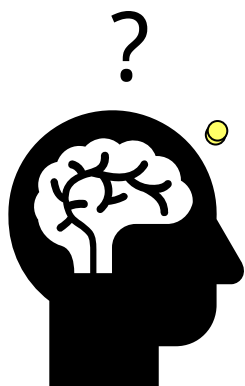
DFDL is used for....

- Scraping data from MS-Word ".doc" files
- Extract metadata of interest from files and skip over the rest
- Image file formats (memory size limitations)
- Parse spreadsheets that are not CSV-like

DFDL not a Transformation Language

What you have....

```
01101001011010  
01010100010101  
00101010010101  
001101111
```



Maybe I can do
this with
DFDL/Daffodil

What you want
is this data....

```
<x>6.847</x>  
<y>abc</y>
```

I'll just annotate
this schema for
the data I want...

Described by this
schema

XML
schema

DFDL not a Transformation Language

What you have....

```
01101001011010  
01010100010101  
00101010010101  
001101111
```

What you want
is this data....

```
<x>6.847</x>  
<y>abc</y>
```

Maybe I can do
this with
DFDL/Daffodil

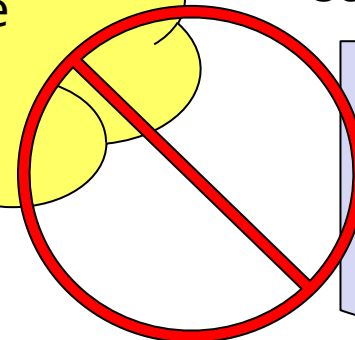
!!!



I'll just annotate
this schema I
found.

Described by this
schema

XML
schema

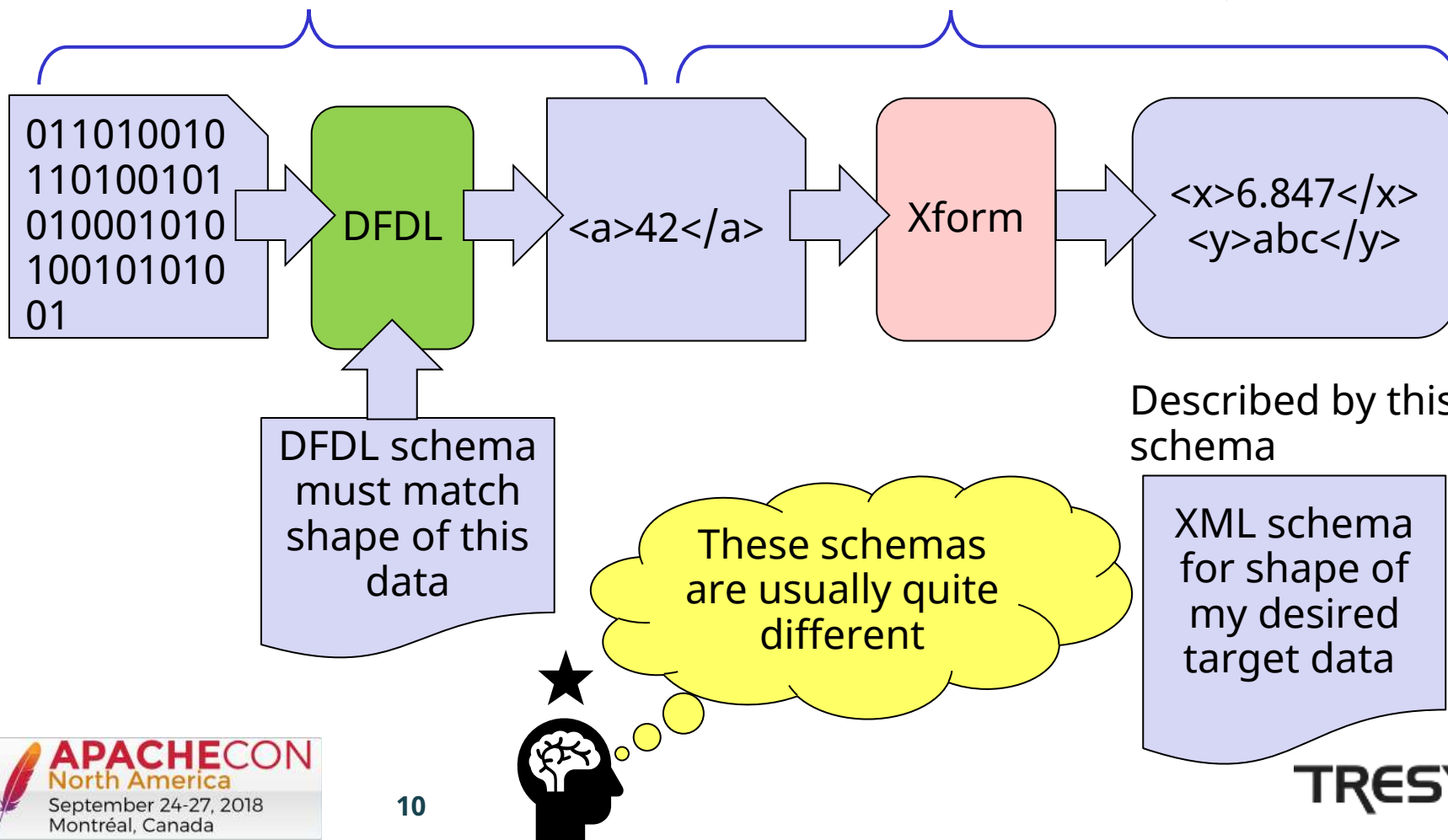


DFDL not a Transformation Language

Describe
the original
file format



Describe how to
transform that into
what you want



DFDL v1.0 is pretty much done

- Closed to new features
 - Any new features are very restricted at this point
- Only Errata/Corrections and clarifications
- Maybe...
a few feature things associated with computed elements
- We're accumulating new feature requests toward DFDL version v2.0 now.

DFDL v2.0 - Wishlist

- Lifting the v1.0 limitations
 - Recursion
 - Offsets
- Layering (implemented already in Daffodil)
- BLOBs (Binary large objects)
 - for image/video files
- Numeric-to-symbolic conversion

How to get a Feature into DFDL

- Propose new DFDL language feature to DFDL Workgroup
 - Email to dfdl-wg@ogf.org
 - Get feedback and refine proposal
- Implement in Daffodil
- Get experience with new feature
 - Refine/improve
 - Create "Experience Document" explaining it
- Formally propose feature for inclusion in DFDL (v2.0)
 - Email to dfdl-wg@ogf.org

Cool Daffodil Ideas...

- Hybrid InfoSet
 - Daffodil directly constructs your native data representation
 - Remove the XML/JSON overheads
- Ultra-fast backend for Daffodil
 - Generate C/C++
 - Generate FPGA for wire-speed parse/unparse

These kinds of things only make sense with a standards-based open source system like Daffodil.

ASN.1

Why is DFDL Needed? - ASN.1 ECN

What about ASN.1 Encoding Control Notation?

- Already an ISO Standard (since 2008)
- Conceptually similar
 - Logical schema language + notations for physical representation
- Very different in the details.
- Developers [Love | Hate] [ASN.1 | XML]

Differences that matter:

- ASN.1 ECN
 - No open source implementation (as of 2018-08-29)
 - Extension of a binary data standard ASN.1 BER/PER/DER
 - Goal to describe legacy protocol messages
- DFDL
 - Open source Daffodil implementation
 - Extension of a textual data standard XML
 - Goal to be union of data integration tool capabilities for format description

Why is DFDL Needed?

Q But what about...

- Apache Avro
- Apache Thrift
- Google GPBs
- ASN.1 BER (or PER/DER/XER)

A Those great, but are *prescriptive*.

They don't describe formats, they *are* data formats themselves.

We need a *descriptive* language.

Why is DFDL Needed?

Q But what about...

ASN.1 ECN (Encoding Control Notation)

A Good point....

- When we started on DFDL, ASN.1 ECN didn't exist yet as a standard. ASN.1 was prescriptive so, not relevant.
- Sources of inspiration for DFDL were existing commercial data integration products. ASN.1 was telco-message centric.
 - DFDL had to be similar to these existing products to be acceptable.
- Once we found out about it, it *seemed* too complex.
 - In the long run this may be naive. DFDL is also complex, as is XML Schema.
- We wanted XML's tangibility (i.e., text) for logical data.
- We were already started down the XML schema path, which was already known to be a preferred approach in the marketplace.
- Internet standards from W3C such as XML and XML schema were hot then, ITU stuff, not so much....

DFDL and ASN.1 ECN

- DFDL
 - Grammar: XML schema (subset)
 - Format descriptions: DFDL annotations
 - Test/Tangibility - XML text
 - Standard: Open Grid Forum (OGF) Proposed Recommendation since 2014
 - OGF is a small, little known organization
 - Commercial - IBM, European Space Agency
 - Open source - Apache Daffodil (Incubating)
- ASN.1 Encoding Control Notation (ECN)
 - Grammar: ASN.1
 - Format Descriptions: Encoding Control Notation
 - Test/Tangibility - ASN.1 Basic Encoding Rules (binary - tools needed to see it)
 - Standard: ITU then ISO Standard since 2008 - ISO/IEC 8825-3
 - ITU organization is big. Many contributors.
 - Commercial - OSS Nokalva
 - Open source - ??? (none for ECN as of my last search 2018-08-29)

Why is DFDL Needed?

- Many other libraries are available
 - Ex: <https://github.com/dloss/binary-parsing>
 - Lists 21 different "binary" tools
 - interestingly... omitting ASN.1 ECN
- Only two are standards-based
 - DFDL - Open Grid Forum (OGF)
 - ASN.1 Encoding Control Notation - ISO/IEC 8825-3
- Of those, only one has an open source implementation
 - DFDL - Daffodil

END

Daffodil Roadmap

✓ 2.1.0 - Current version as of May 2018

Roadmap is maintained on project wiki

<https://cwiki.apache.org/confluence/display/DAFFODIL/Apache+Daffodil>

- 2.2.0 - Two new important features
 - ✓ Layer support - enables line folding, base64, compress, ...
 - ✓ Message streaming API - enables unbounded streams of messages
- 2.3.0 - Interoperability
 - Features to enable Daffodil to run more IBM-published DFDL schemas
 - Cross-validation test system
 - BLOB support - enables large image file formats
- 2.4.0 - Usability, Community Growth
 - Debug/Trace improvements
 - Schema compilation speed/space improvements
 - Developer documentation of Daffodil's internals.
 - SAX-style event streaming - large documents/files (non-BLOB)
 - Tutorials on DFDL
 - ... *and your needed features here!*

Daffodil Performance

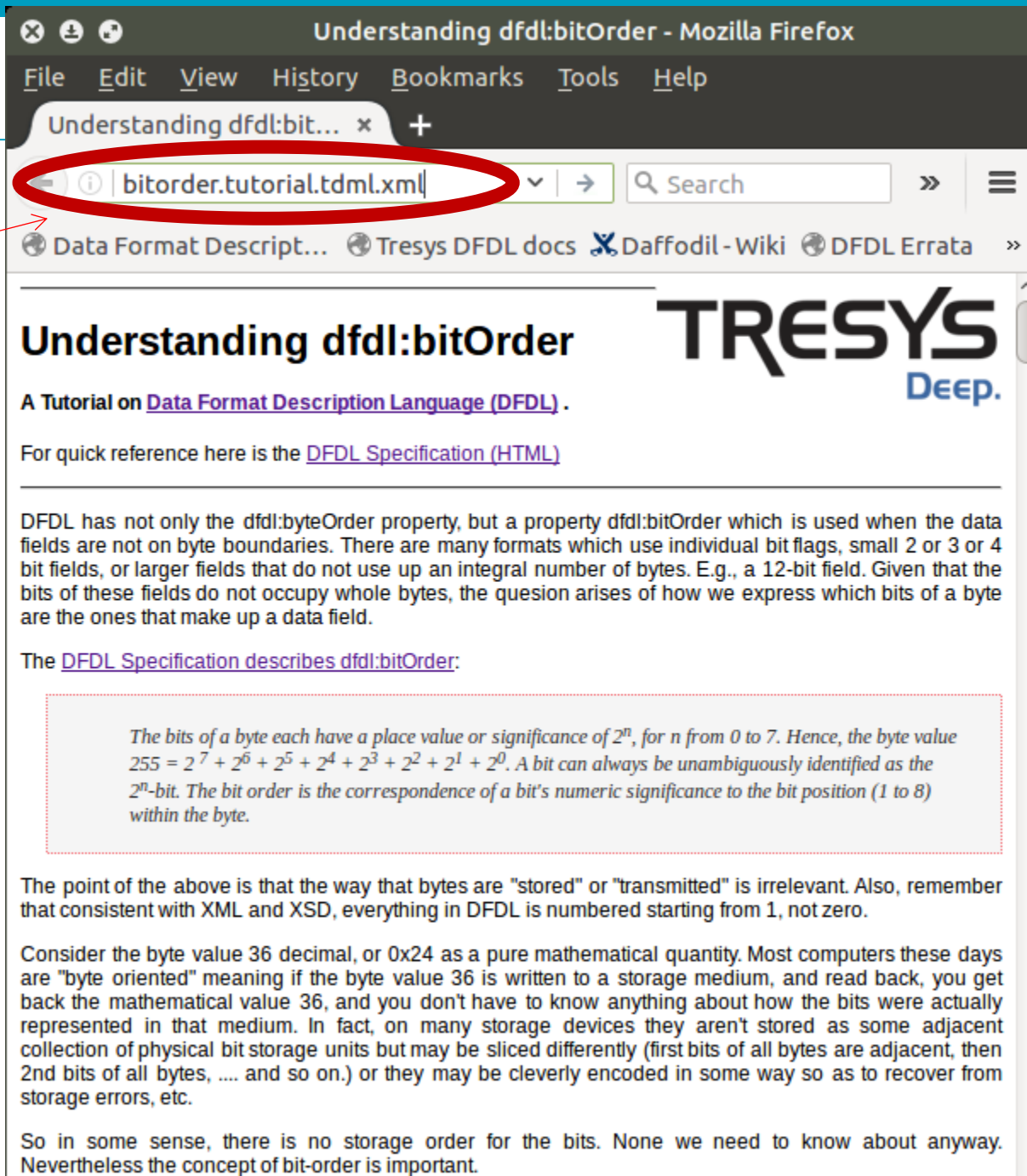
Varies with complexity of data format

Single-thread timings, as of 2017-09-01, Daffodil 2.0.0 release

format	parse rate (MB/sec)	unparse rate (MB/sec)
CSV - simple delimited text	1.2	2.8
Link16 J3.5 - dense binary	0.8 *	0.2
PCAP - Binary IP Packets	20	2.2
ATO - complex delimited text	0.3	0.9

* = Approximately 20K Link16 NACT messages/sec with 3 link16 words each. (~60 bytes/msg)

HTML page
rendered from
“.tdml.xml” file
directly by
Firefox using
XSLT and CSS



Understanding dfdl:bitOrder - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Understanding dfdl:bit... x +

bitorder.tutorial.tdml.xml

Data Format Descript... Tresys DFDL docs Daffodil - Wiki DFDL Errata

Understanding dfdl:bitOrder

TRESYS
Deep.

A Tutorial on [Data Format Description Language \(DFDL\)](#) .

For quick reference here is the [DFDL Specification \(HTML\)](#)

DFDL has not only the dfdl:byteOrder property, but a property dfdl:bitOrder which is used when the data fields are not on byte boundaries. There are many formats which use individual bit flags, small 2 or 3 or 4 bit fields, or larger fields that do not use up an integral number of bytes. E.g., a 12-bit field. Given that the bits of these fields do not occupy whole bytes, the question arises of how we express which bits of a byte are the ones that make up a data field.

The [DFDL Specification describes dfdl:bitOrder](#):

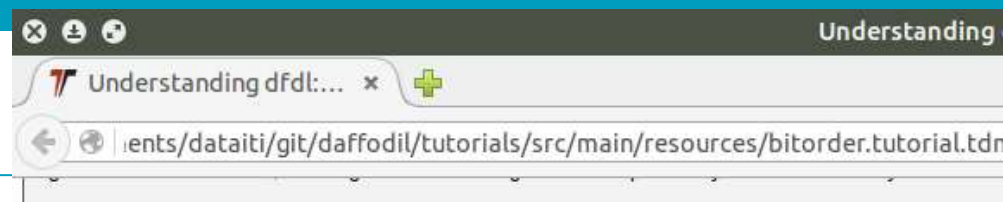
The bits of a byte each have a place value or significance of 2^n , for n from 0 to 7. Hence, the byte value $255 = 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$. A bit can always be unambiguously identified as the 2^n -bit. The bit order is the correspondence of a bit's numeric significance to the bit position (1 to 8) within the byte.

The point of the above is that the way that bytes are "stored" or "transmitted" is irrelevant. Also, remember that consistent with XML and XSD, everything in DFDL is numbered starting from 1, not zero.

Consider the byte value 36 decimal, or 0x24 as a pure mathematical quantity. Most computers these days are "byte oriented" meaning if the byte value 36 is written to a storage medium, and read back, you get back the mathematical value 36, and you don't have to know anything about how the bits were actually represented in that medium. In fact, on many storage devices they aren't stored as some adjacent collection of physical bit storage units but may be sliced differently (first bits of all bytes are adjacent, then 2nd bits of all bytes, and so on.) or they may be cleverly encoded in some way so as to recover from storage errors, etc.

So in some sense, there is no storage order for the bits. None we need to know about anyway. Nevertheless the concept of bit-order is important.

TDML test parts are
also rendered



Let's run this example. Here's the schema:

DFDL Schema:

```
1.
2.   <dfdl:format ref="ex:daffodilTest1" bitOrder="mostSignificantBitFirst"
3.
4.   <xs:element name="mostFirst" dfdl:lengthKind="implicit">
5.     <xs:complexType>
6.       <xs:sequence>
7.         <xs:element name="A" type="xs:int" dfdl:length="3" />
8.         <xs:element name="B" type="xs:int" dfdl:length="7" />
9.         <xs:element name="C" type="xs:int" dfdl:length="4" />
10.        <xs:element name="D" type="xs:int" dfdl:length="2" />
11.      </xs:sequence>
12.    </xs:complexType>
13.  </xs:element>
14.
15.
```

Notice the above schema specifies mostSignificantBitFirst. This applies to all the elements within this schema.

The data in hex is:

Data Stream:

1.	
2.	62 55
3.	

And the infoset we get when we parse is what we expected.

Infoset:

```
1. <tdml:infoset>
2.   <tdml:dfdlInfoset>
3.     <mostFirst>
4.       <A>3</A>
5.       <B>9</B>
6.       <C>5</C>
7.       <D>1</D>
8.     </mostFirst>
```

Daffodil Unparser dfdl:outputValueCalc

- DFDL and Daffodil have powerful computation capabilities not seen in other prior format description systems.
- Unparser can compute element values to store earlier in the data, based on infoSet elements that are later.
- InfoSet events are received, infoSet tree is built incrementally.
- outputValueCalc element nodes are added to tree.
- Producer "Co-routine" (really just an object) for expression evaluation is queued on nodes waiting for values or children.
- Data output streams start direct, split off a buffering part, then are collapsed back.