Data Format Description Language (DFDL)

& the

Apache Daffodil™ Project

DFDL is ISO/IEC 23415

DFDL is OGF GFD.240

For Daffodil Version 4.0.0

2025-11-11

Agenda

- Motivation: Why is DFDL Needed?
- Introduction to DFDL
  - Use Cases for DFDL
- Examples
  - CSV, PCAP, MIL-STD-2045
- DFDL Schemas
- Apache Daffodil Status
- Example
  - Daffodil Java API Use

Motivation

Why is DFDL needed?

Why ISO/OGF DFDL?

There are *hundreds* of ad-hoc data format description systems

Every Enterprise Software Company

- IBM (10+)
- Oracle(10+)
- SAP(10+)
- Microsoft
- SAS
- SyncSort
- AbInitio
- Pervasive
- Qlik/Expressor
- …. Dozens more

Every kind of software that takes in data:

- data directed routing
- database
- data analysis and/or data mining
- data cleansing
- master data management
- application integration
- data visualization

All these data format descriptions are:

- ***proprietary***
- ***ad-hoc***
- ***incompatible***

Even within products of the same company!

Why DFDL is Needed

- Avoid customer lock in by vendors

- Provide skills leverage
  - DFDL is an ISO standard
  - DFDL has an Open-Source implementation

Why is DFDL Needed?

- But what about…
  - Apache Avro
  - Apache Thrift
  - Google Protocol Buffers
  - ASN.1 BER (or PER/DER/XER)
- Those are great but are *prescriptive*.
  - They don't describe formats; they *are* data formats
- We need a *descriptive* language.

Why is DFDL Needed? - ASN.1 ECN

- What about ASN.1 Encoding Control Notation?
- Doesn't it do what DFDL Does?

- ASN.1 ECN is *Conceptually* Similar
  - Logical schema language
  - Separate file of ECN notations for physical representation
  - Also ISO Standard (since 2008)

- ASN.1 ECN has Very Different Capabilities
  - Does not support textual data formats
  - No current open-source implementation of ECN (as of 2025)

- Want a side-by-side comparison?
  - See the DFDLSchemas FakeTDL DFDL Schema project

Solving the Data Format Problem

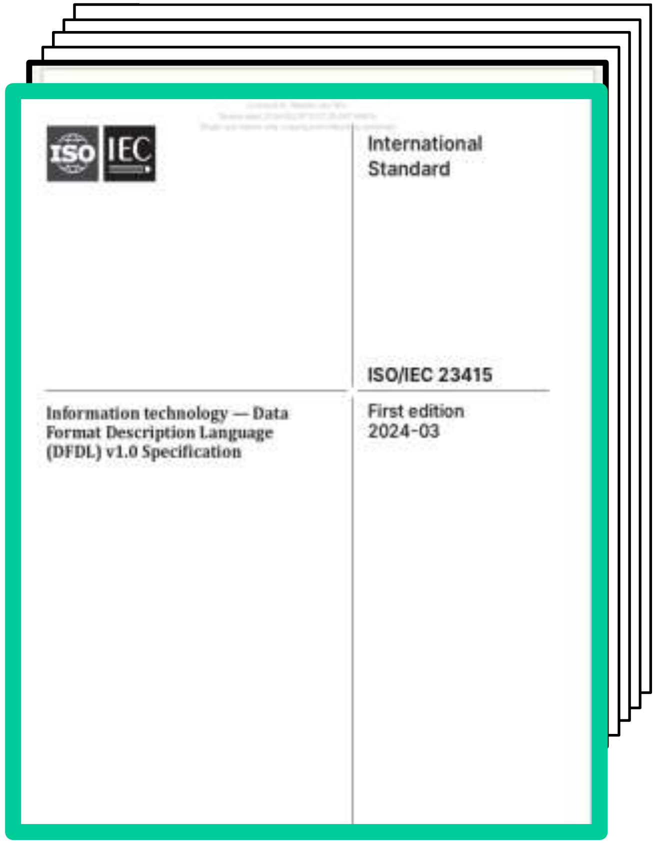
Three things are required:

- Open Standard DFDL
    - Long-term sponsors
        - IBM – has commercial DFDL implementations
        - US DoD, Canada DND
    - ISO/IEC and OGF Standard
- DFDL schemas for important data formats
- A High-Quality Open-Source Library Implementation
    - With a supporting community of developers
    - With available commercial support

INTRODUCTION TO DFDL

DFDL is ISO/IEC 23415:2024

- ISO Publicly Available Standard (PAS)
  - PAS - Pairs a small standards org with ISO
  - Lots of small standard orgs do this: OGF, W3C, DMTF, OMG, …
- You can still get it free
  - from Open Grid Forum (OGF) as GFD.240
- Future OGF versions will become ISO PAS



International Standard

ISO/IEC 23415

Information technology — Data Format Description Language (DFDL) v1.0 Specification

First edition
2024-03

Data Format Description Language

- DFDL is a way of describing data formats
- It is NOT a data format itself!

- DFDL combines State-of-the-Art
- Union of capabilities across many marketplace data integration products/tools/packages

- DFDL adds small number of real innovations
- Overcome limitations of prior-gen e.g.,
    - Computed Elements Capability - Unparse as well as Parse!
    - Expression language
    - BitOrder

Data Format Description Language

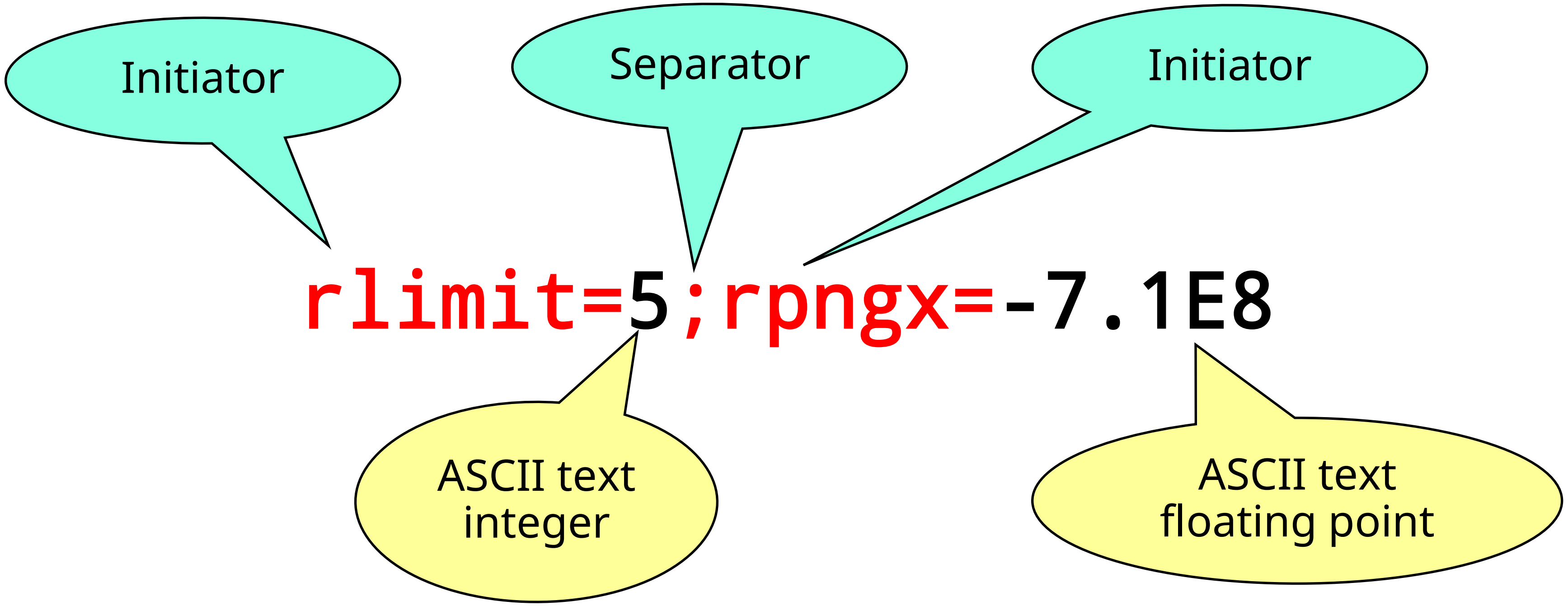
- Core Concepts
  - Leverage XML Schema (XSDL)
    - Used as the Grammar scaffolding
    - Describes the logical data model
    - DFDL uses only a subset of XML schema
    - XSDL provides standard ways to annotate schemas. DFDL uses this.
  - Add DFDL annotations
    - Describe the physical representation.
  - Read and write from same DFDL Schema

- Because Developers [Love | Hate] XML
  - The DFDL Schema is based on XSDL
  - But … the Infoset created when parsing data
    - Can be XML
    - Can be JSON
    - Via API - can directly populate ex: Apache Drill Rows, Apache NiFi Records

Example – Delimited Text Data

```
rlimit=5;rpngx=-7.1E8
```

Example – Delimited Text Data



`rlimit=5;rpngx=-7.1E8`

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

DFDL Schema - Logical

Logical Elements

```
<xs:complexType name=“rPair">
  <xs:sequence>
       <xs:element name=“rlim" type="xs:int"/>

     <xs:element name=“rpng" type="xs:float"/>

  </xs:sequence>
</xs:complexType>
```

DFDL schema - Physical

```
<xs:annotation>
 <xs:appinfo source="http://www.ogf.org/dfdl/">
   <dfdl:format representation="text"
        textNumberRep="standard" encoding="ascii"
        lengthKind="delimited“ .../>
  </xs:appinfo>
</xs:annotation>

<xs:complexType name=“rPair”>
 <xs:sequence dfdl:separator=";" ... >
  <xs:element name="rLim" type="xs:int"
          dfdl:initiator=“rLimit=" ... />
   <xs:element name=“rpng" type="xs:float"
          dfdl:initiator=“rpngx=" ... />
 </xs:sequence>
</xs:complexType>
```

Top level format declaration block applies to this entire schema *file*.

DFDL properties

;

rLimit=5

rpngx=-7.1E8

DFDL schema *is also* an XML Schema

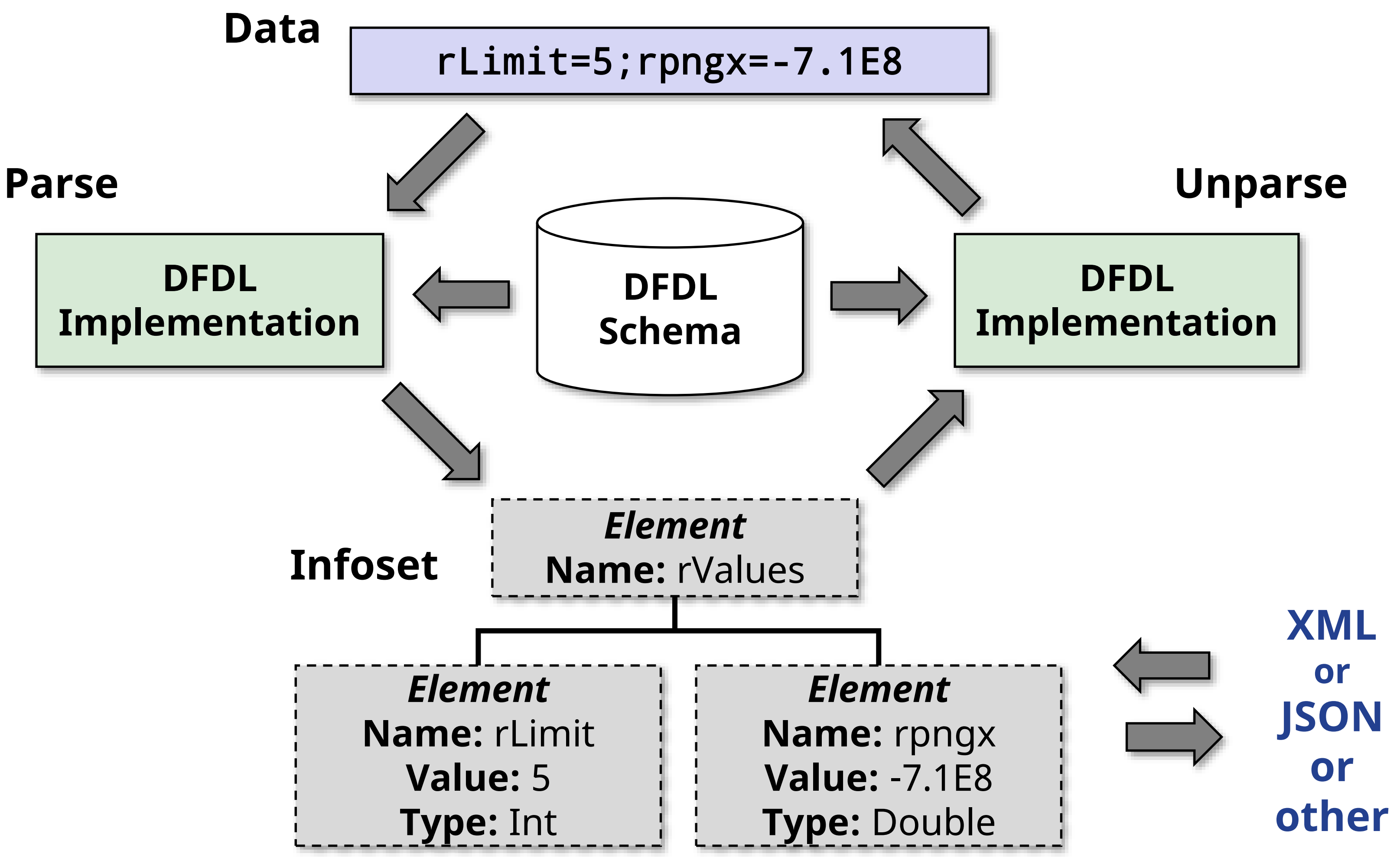
```
<xs:complexType name=“rPair">
  <xs:sequence dfdl:separator=";" ... >
    <xs:element name="rLim" type="lt100"
                dfdl:initiator=“rLimit=" ... />
     <xs:element name=“rpng" type="xs:float"
                dfdl:initiator=“rpngx=" ... />
  </xs:sequence>
</xs:complexType>

<simpleType name="lt100">
   <restriction base="xs:int">
     <maxInclusive value="100"/>
   </restriction>
</simpleType>
```

Type reference

XML Schema facet constraints are also part of the DFDL schema

A good DFDL schema has both the DFDL properties AND the XSD facet constraints.

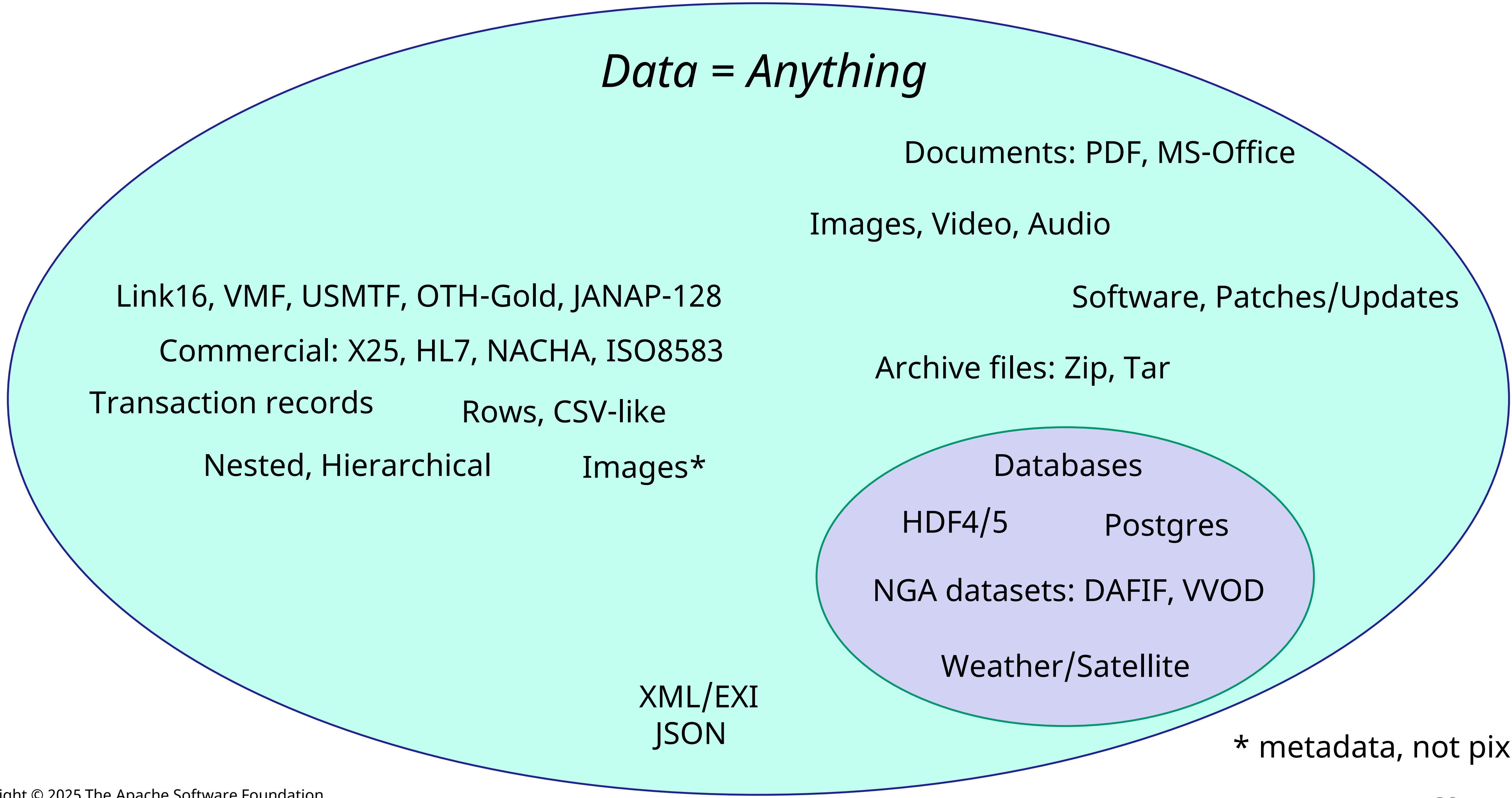
DFDL Data and Infoset Lifecycle



DFDL Use Cases

- Deep Content Inspection/Sanitization
  - Cross Domain Solutions, Network Guards
  - Data Filtering
  - Rip-and-Rebuild Firewalls
- Data-Directed Routing
- Data Intake
  - Data Warehouse or Analytical Data Store
- Interoperability / Data Translation
- Conversion of Legacy Data to Modern Formats
- Open Data / Data Export
  - Recasting data as XML or JSON for export/publishing

Q: What Kinds of Data is DFDL Good For?
A: Not everything



Data = Anything

Documents: PDF, MS-Office

Images, Video, Audio

Link16, VMF, USMTF, OTH-Gold, JANAP-128

Software, Patches/Updates

Commercial: X25, HL7, NACHA, ISO8583

Archive files: Zip, Tar

Transaction records

Rows, CSV-like

Nested, Hierarchical

Images*

Databases

HDF4/5

Postgres

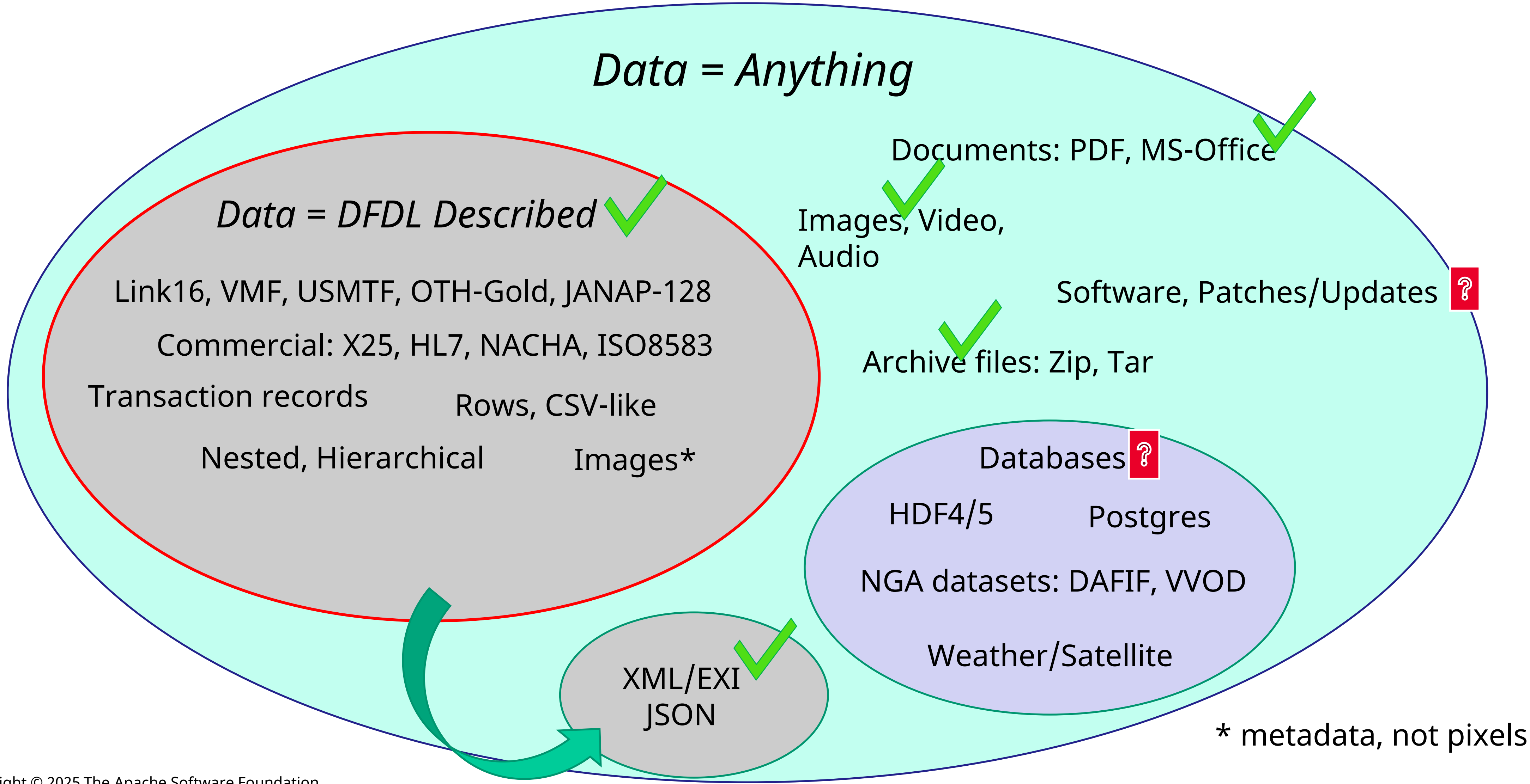
NGA datasets: DAFIF, VVOD

Weather/Satellite

XML/EXI
JSON

* metadata, not pixels

Q: What Kinds of Data is DFDL Good For?
A: Not everything



Data = Anything

Data = DFDL Described ✓

Link16, VMF, USMTF, OTH-Gold, JANAP-128

Commercial: X25, HL7, NACHA, ISO8583

Transaction records

Rows, CSV-like

Nested, Hierarchical

Images*

Documents: PDF, MS-Office ✓

Images, Video, Audio ✓

Software, Patches/Updates ?

Archive files: Zip, Tar ✓

Databases ?

HDF4/5

Postgres

NGA datasets: DAFIF, VVOD

Weather/Satellite

XML/EXI
JSON ✓

* metadata, not pixels

Current DFDL v1.0 Language Limitations

- Recursive types
  - DFDL v1.0 not a Turing-Complete language
  - On purpose - it's a feature, not a bug

- Position of elements "by offset"
  - Random jumping around data
  - Ex: TIFF file format
    - TIFF cannot be described in DFDL v1.0

Thanks to
http://langsec.org/occupy/

Things DFDL (v1.0) Doesn't Do

- DFDL v1.0 was not originally intended for:
  - Document file formats
    - e.g., MS Office documents (.doc), or RTF, PDF
  - Images, Video, Audio
  - Archives like zip files or tar files
  - Core dump/memory image format
  - Storage format of a RDBMS table
  - Graphs of pointers - object graphs dumped from memory

- DFDL is being extended to cover more
  - BLOBs - for images, video, audio
    - Already available in Daffodil from v2.5.0

Examples

CSV (ASCII text delimited)

PCAP (variable length binary)

MIL-STD-2045 (dense bit-packed binary, LSBF)

Example: CSV

Comma-Separated Values

Example - CSV

- Textual Format (ASCII)
- Comma-Separated Values
- Tabular data
- Rows separated by newlines
- Fields separated by commas

```
Smith,Charles,36,6.2
Williams,Mary,51,5.5
```

Example - CSV

```
<schema>
  <annotation>
    <appinfo source="http://www.ogf.org/dfdl/">
       <dfdl:format ref="defaults"
                   representation="text"
                   encoding="ASCII"
                   lengthKind="delimited"
                   escapeSchemRef="quoted"
                   occursCountKind="parsed" />
      <dfdl:defineEscapeScheme name="quoted">
         <dfdl:escapeScheme escapeKind="escapeBlock"
                   escapeBlockStart="&quot;"
                   escapeBlockEnd="&quot;"
                   escapeEscapeCharacter="&quot;">
      </dfdl:defineEscapeScheme>
    </appinfo>
  <annotation>


<schema>
```

Example - CSV

```
<schema>
  <annotation>
    <appinfo source="http://www.ogf.org/dfdl/">
      <dfdl:format ref="defaults"
                   representation="text"
                   encoding="ASCII"
                   lengthKind="delimited"
                   escapeSchemRef="quoted"
                   occursCountKind="parsed" />
      <dfdl:defineEscapeScheme name="quoted">
         <dfdl:escapeScheme escapeKind="escapeBlock"
                   escapeBlockStart="&quot;"
                   escapeBlockEnd="&quot;"
                   escapeEscapeCharacter="&quot;">
      </dfdl:defineEscapeScheme>
    </appinfo>
  <annotation>


<schema>
```

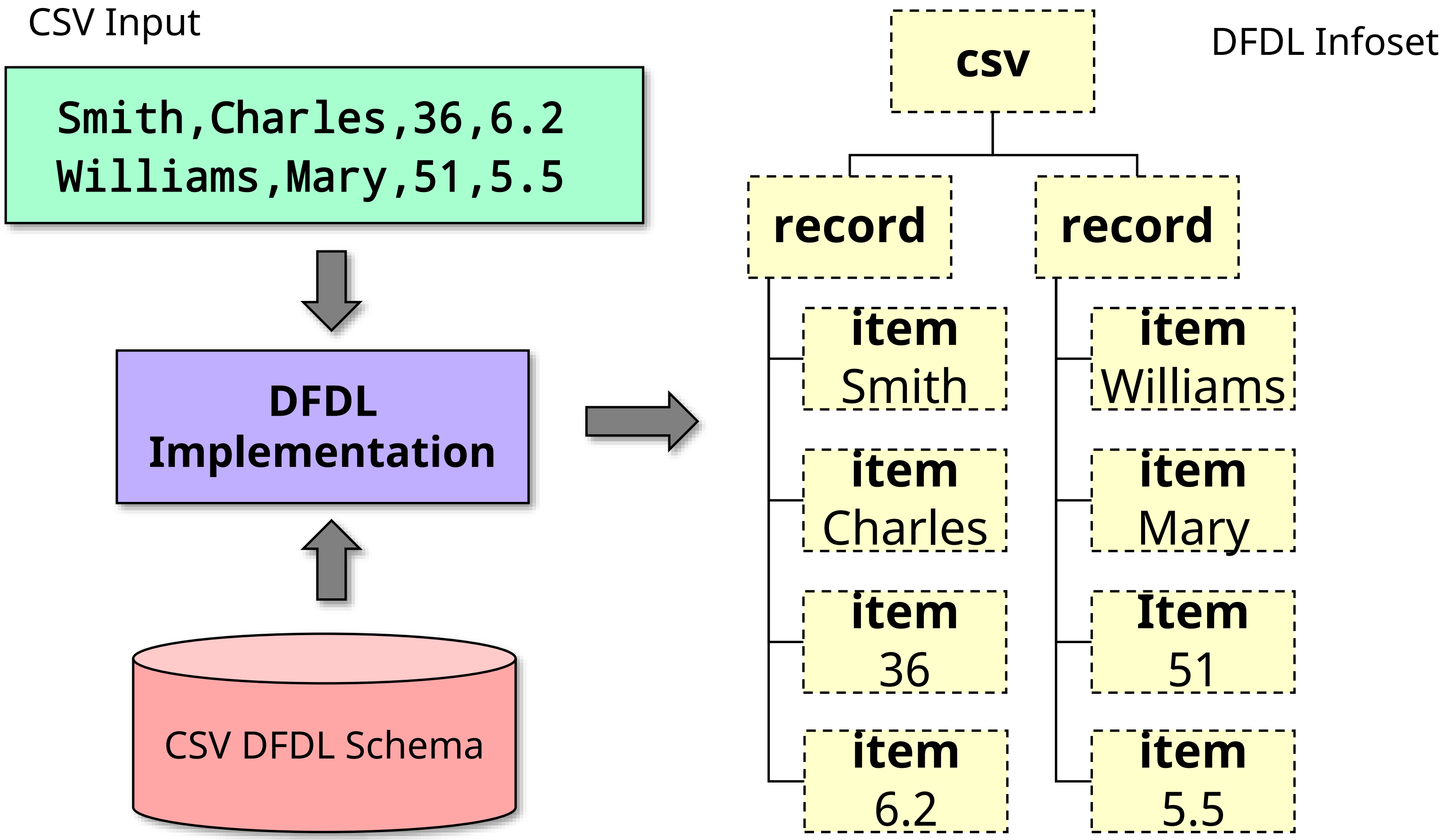
Example - CSV

```
<element name="csv">
  <complexType>
    <sequence dfdl:separator="%NL;"
                 dfdl:separatorPosition="postfix">
      <element name="record" maxOccurs="unbounded">
        <complexType>
          <sequence dfdl:separator=","
                       dfdl:separatorPosition="infix">
            <element name="item" type="xs:string"
                        maxOccurs="unbounded"/>
          </sequence>
        </complexType>
      </element>
    </sequence>
  </complexType>
</element>
```

Example - CSV

```
<element name="csv">
  <complexType>
    <sequence dfdl:separator="%NL;"
                 dfdl:separatorPosition="postfix">
      <element name="record" maxOccurs="unbounded">
        <complexType>
          <sequence dfdl:separator=","
                       dfdl:separatorPosition="infix">
            <element name="item" type="xs:string"
                        maxOccurs="unbounded"/>
          </sequence>
        </complexType>
      </element>
    </sequence>
  </complexType>
</element>
```

Example - CSV



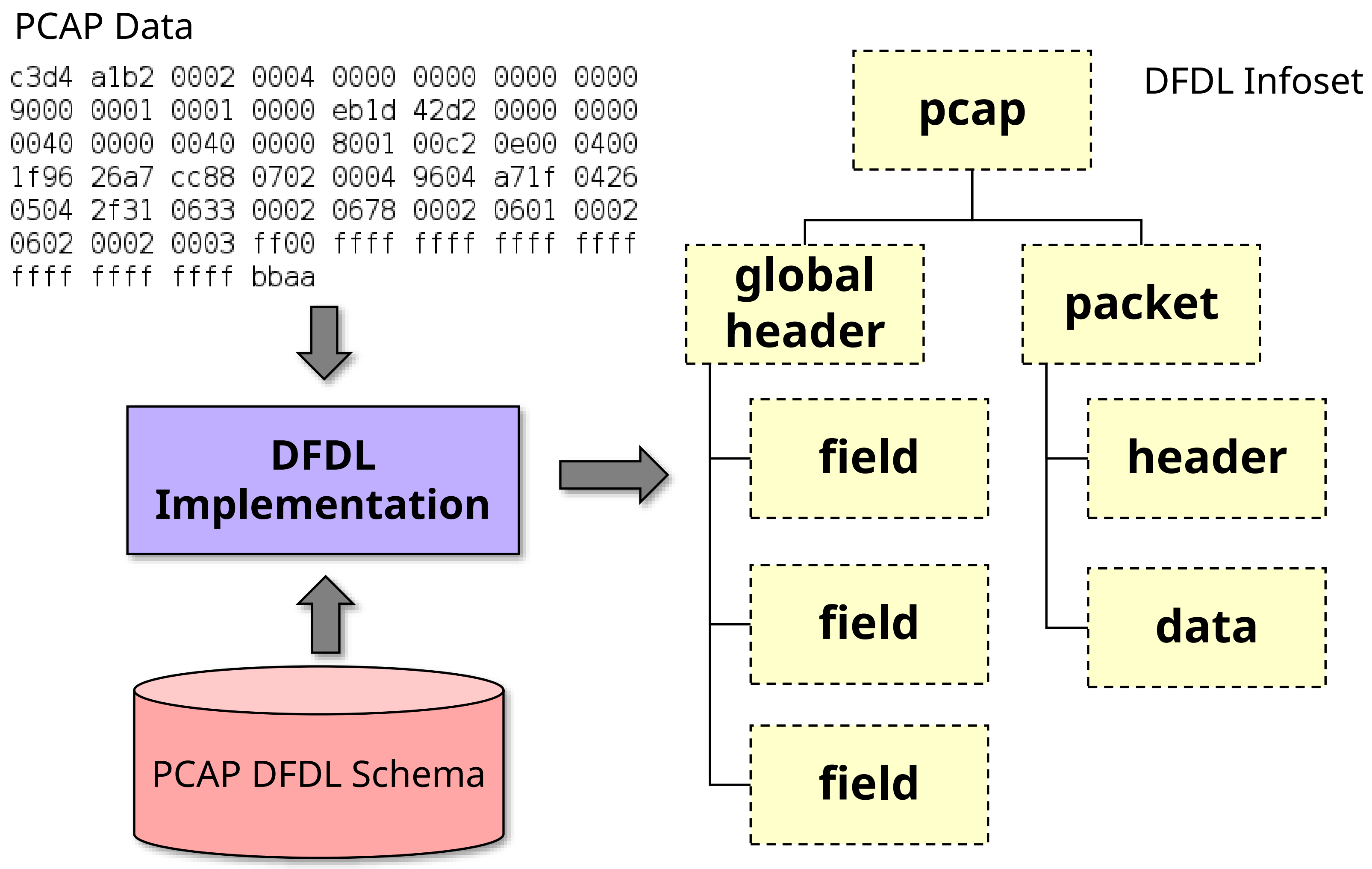
CSV Input

```
Smith,Charles,36,6.2
Williams,Mary,51,5.5
```

Example: PCAP

Packet Capture File Format

Example - PCAP

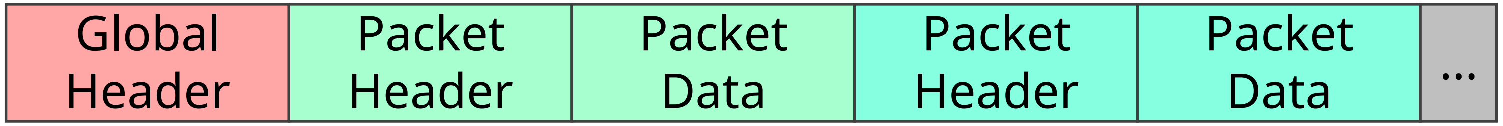


PCAP Data

```
c3d4 a1b2 0002 0004 0000 0000 0000 0000
9000 0001 0001 0000 eb1d 42d2 0000 0000
0040 0000 0040 0000 8001 00c2 0e00 0400
1f96 26a7 cc88 0702 0004 9604 a71f 0426
0504 2f31 0633 0002 0678 0002 0601 0002
0602 0002 0003 ff00 ffff ffff ffff ffff
ffff ffff ffff bbaa
```

Example - PCAP

- Network Packet Capture
- Binary Format
- Global Header
- Packet Header/Data
- Variable Endianness and Data Length



Global Header	Packet Header	Packet Data	Packet Header	Packet Data	...

Example - PCAP

```
<schema>
  <annotation>
    <appinfo source="http://www.ogf.org/dfdl/">
      <dfdl:format ref="defaults"
                   representation="binary"
                   alignment="8"
                   alignmentUnits="bits"
                   lengthUnits="bits"
                   binaryNumberRep="binary"
                   lengthKind="implicit"
                   byteOrder="{ $byte_order }" />
      <dfdl:defineVariable name="byte_order" type="xs:string"/>
    </appinfo>
  </annotation>

  <simpleType name="uint32" dfdl:lengthKind="explicit"
                               dfdl:length="32">
    <restriction base="xs:unsignedInt"/>
  </simpleType>
</schema>
```

Example - PCAP

```
<schema>
  <annotation>
    <appinfo source="http://www.ogf.org/dfdl/">
      <dfdl:format ref="defaults"
                   representation="binary"
                   alignment="8"
                   alignmentUnits="bits"
                   lengthUnits="bits"
                   binaryNumberRep="binary"
                   lengthKind="implicit"
                   byteOrder="{ $byte_order }" />
      <dfdl:defineVariable name="byte_order" type="xs:string"/>
    </appinfo>
  </annotation>

  <simpleType name="uint32" dfdl:lengthKind="explicit"
                               dfdl:length="32">
    <restriction base="xs:unsignedInt"/>
  </simpleType>
</schema>
```

Example - PCAP

```
<schema>
  <annotation>
    <appinfo source="http://www.ogf.org/dfdl/">
      <dfdl:format ref="defaults"
                   representation="binary"
                   alignment="8"
                   alignmentUnits="bits"
                   lengthUnits="bits"
                   binaryNumberRep="binary"
                   lengthKind="implicit"
                   byteOrder="{ $byte_order }" />
      <dfdl:defineVariable name="byte_order" type="xs:string"/>
    </appinfo>
  </annotation>

  <simpleType name="uint32" dfdl:lengthKind="explicit"
                               dfdl:length="32">
    <restriction base="xs:unsignedInt"/>
  </simpleType>
</schema>
```

Example - PCAP

```
<element name="magic_number" type="ex:uint32"
            dfdl:byteOrder="bigEndian">
  <annotation>
    <appinfo source="http://www.ogf.org/dfdl/">
      <dfdl:setVariable ref="byte_order"><![CDATA[{
        if (. eq dfdl:unsignedInt('xA1B2C3D4')) then 'bigEndian'
        else if (. eq dfdl:unsignedInt('xD4C3B2A1') then 'littleEndian'
        else daf:error() }]]>
      </dfdl:setVariable>
      <dfdl:property name="outputValueCalc"><![CDATA[{
           if ($dfdl:byteOrder eq 'bigEndian')
           then dfdl:unsignedInt('xA1B2C3D4')
           else dfdl:unsignedInt('xD4C3B2A1')
          }]]></dfdl:property>

    </appinfo>
  </annotation>
</element>
```

Example - PCAP

```
<element name="magic_number" type="ex:uint32"
            dfdl:byteOrder="bigEndian">
  <annotation>
    <appinfo source="http://www.ogf.org/dfdl/">
      <dfdl:setVariable ref="byte_order"><![CDATA[{
        if (. eq 2712847316) then 'bigEndian'
        else if (. eq 3569595041) then 'littleEndian'
        else daf:error() }]]>
      </dfdl:setVariable>
      <dfdl:property name="outputValueCalc"><![CDATA[{
           if ($dfdl:byteOrder eq 'bigEndian')
           then dfdl:unsignedInt('xA1B2C3D4')
           else dfdl:unsignedInt('xD4C3B2A1')
           }]]></dfdl:property>

    </appinfo>
  </annotation>
</element>
```

Example - PCAP

```
<element name="PacketHeader">
   <complexType>
      <sequence>
         <element name="Seconds" type="pcap:uint32"/>
         <element name="USeconds" type="pcap:uint32"/>
         <element name="InclLen" type="pcap:uint32"
           ...
         />
         <element name="OrigLen" type="pcap:uint32"
            ...
          />
      </sequence>
   </complexType>
</element>
...
<element ref="pcap:LinkLayer"
   dfdl:lengthUnits="bytes" dfdl:lengthKind="explicit"
   dfdl:length="{ ../PacketHeader/InclLen }"/>
```

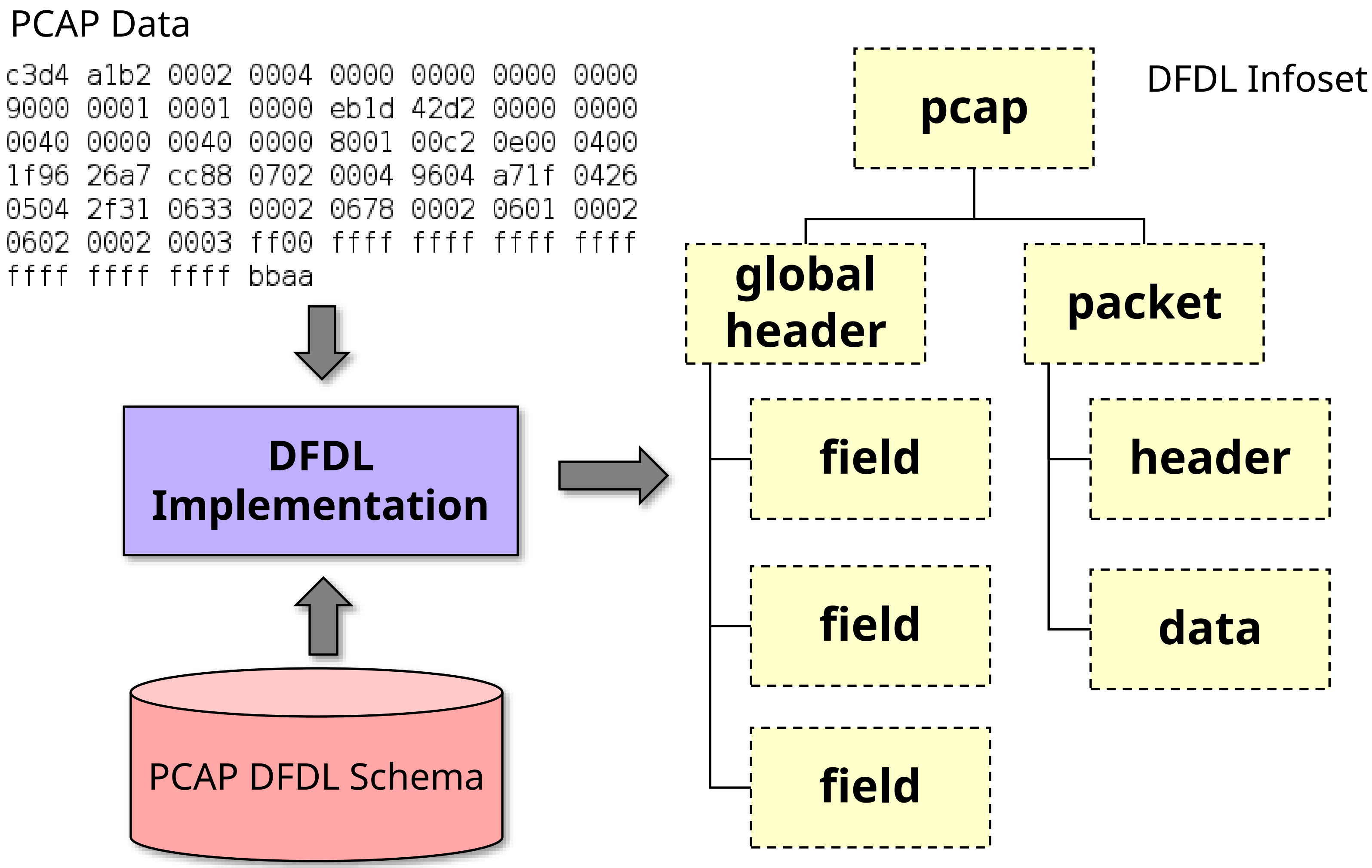
Example - PCAP

```
<element name="PacketHeader">
   <complexType>
      <sequence>
         <element name="Seconds" type="pcap:uint32"/>
         <element name="USeconds" type="pcap:uint32"/>
         <element name="InclLen" type="pcap:uint32"
           ...
         />
         <element name="OrigLen" type="pcap:uint32"
            ...
          />
      </sequence>
   </complexType>
</element>
...
<element ref="pcap:LinkLayer"
   dfdl:lengthUnits="bytes" dfdl:lengthKind="explicit"
   dfdl:length="{ ../PacketHeader/InclLen }"/>
```

Example - PCAP

```
<element name="PacketHeader">
   <complexType>
      <sequence>
         <element name="Seconds" type="pcap:uint32"/>
         <element name="USeconds" type="pcap:uint32"/>
         <element name="InclLen" type="pcap:uint32"
           dfdl:outputValueCalc="{
            if (dfdl:valueLength(
                      ../../pcap:LinkLayer/pcap:Ethernet,
                      'bytes') le 60) then 60
            else
               dfdl:valueLength(
                      ../../pcap:LinkLayer/pcap:Ethernet,
                      'bytes') }"
         />
         <element name="OrigLen" type="pcap:uint32"
            dfdl:outputValueCalc="{ ../InclLen }"/>
      </sequence>
   </complexType>
</element>
...
<element ref="pcap:LinkLayer"
   dfdl:lengthUnits="bytes" dfdl:lengthKind="explicit"
   dfdl:length="{ ../PacketHeader/InclLen }"/>
```

Example - PCAP



PCAP Data

```
c3d4 a1b2 0002 0004 0000 0000 0000 0000
9000 0001 0001 0000 eb1d 42d2 0000 0000
0040 0000 0040 0000 8001 00c2 0e00 0400
1f96 26a7 cc88 0702 0004 9604 a71f 0426
0504 2f31 0633 0002 0678 0002 0601 0002
0602 0002 0003 ff00 ffff ffff ffff ffff
ffff ffff ffff bbaa
```

Example: MIL-STD-2045

A dense bit-packed binary data format similar to many MIL-STD formats, but publicly available.

Uses Least-Significant-Bit-First bit order – typical of MIL formats, different from most non-MIL formats.

Example - MIL-STD-2045

- Densely packed binary with unique bit order

```
dfdl:representation="binary"
dfdl:alignment="1"
dfdl:alignmentUnits="bits"
dfdl:lengthUnits="bits"
dfdl:lengthKind="explicit"
dfdl:length
dfdl:byteOrder="littleEndian"
dfdl:bitOrder="leastSignificantBitFirst"
```

Example - MIL-STD-2045

- 7-bit strings embedded in binary

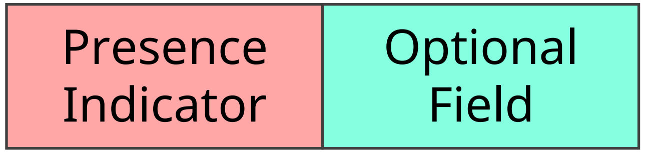
```
dfdl:encoding=“X-DFDL-US-ASCII-7BIT-PACKED"
```

□ 0x7F (DEL) terminated, unless max-length met

```
<element name="str50" type="xs:string"
            dfdl:lengthPattern="[^\x7F]{0,49}(?=\x7F)|.{50}" />

<sequence dfdl:terminator="{
  if (fn:string-length(./str50) eq 50)
  then '%ES;'
  else '%DEL;'
}" />
```

Example - MIL-STD-2045

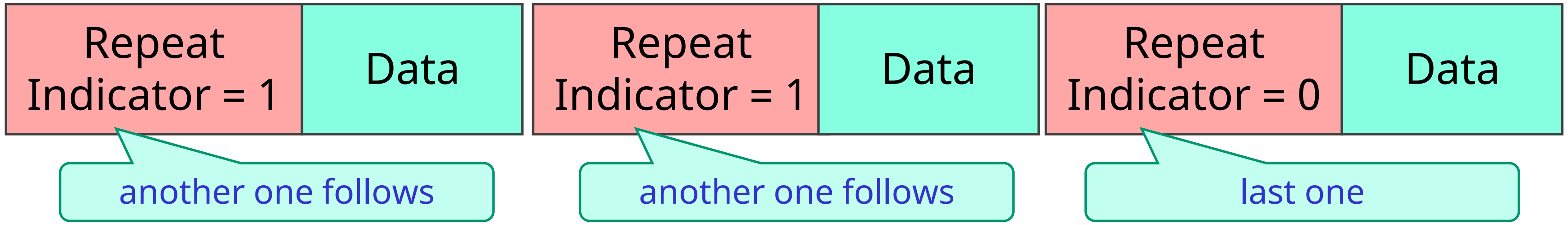
- Presence indicators



```
<dfdl:discriminator test="{ . eq 1 }" />
```

Example - MIL-STD-2045

- Repeat indicators



```
dfdl:occursCountKind="implicit"
minOccurs="1"
maxOccurs="unbounded"

<dfdl:discriminator test="{
  if (dfdl:occursIndex() eq 1)
  then fn:true()
  else ../fields[dfdl:occursIndex() – 1]/repeat_indicator eq 1 }" />

 <element name="GRI" type="tns:repeatIndicator"
   dfdl:outputValueCalc="{
     if (dfdl:occursIndex() lt fn:count(..))
     then 1 else 0 }" />
```

More DFDL Properties

```
dfdl:assert
dfdl:discriminator
dfdl:newVariableInstance
dfdl:setVariable
dfdl:defineEscapeSchema
dfdl:defineVariable
dfdl:ignoreCase
dfdl:encodingErrorPolicy
dfdl:alignment
dfdl:alignmentUnits
dfdl:fillByte
dfdl:leadingSkip
dfdl:trailingSkip
dfdl:emptyValueDelimiterPolicy
dfdl:outputNewLine
dfdl:prefixIncludesPrefixLength
dfdl:prefixLengthType
dfdl:lengthPattern
dfdl:textPadKind
dfdl:textTrimKind
dfdl:textOutputMinLength
dfdl:escapeSchemeRef
dfdl:escapeKind
dfdl:escapeCharacter
dfdl:escapeBlockStart
dfdl:escapeBlockEnd
dfdl:escapeEscapeCharacter
dfdl:extraEscapedCharacters
dfdl:generateEscapeBlock
dfdl:textBidi
dfdl:textStringJustification
dfdl:textStringPadCharacter
dfdl:textNumberJusitification
dfdl:textStandardNaNRep
dfdl:textStandardZeroRep
dfdl:textStandardBase
dfdl:binaryNumberRep
dfdl:binaryPackedSignCodes
dfdl:binaryFloatRep
dfdl:calendarPattern
dfdl:binaryCalendarRep
dfdl:nilValueDelimiterPolicy
dfdl:initiatedContent
dfdl:separatorPosition
dfdl:separatorSuppressionPolicy
dfdl:floating
dfdl:hiddenGroupRef
dfdl:initiatedContent
dfdl:occursCountKind
dfdl:occursStopValue
dfdl:inputValueCalc
dfdl:outputValueCalc
```

DFDL Schemas

What's available? Where to find them?

DFDL Schemas - Free/Open-Source

- Github DFDLSchemas
  - many useful learning examples
    - fakeTDL
    - envelope-payload
    - tcpHeader
    - PCAP
    - ethernetIP
  - many real production-use schemas
    - MIL-STD-2045 header schema
    - syslog
    - shape
    - NITF
    - MagVar
  - commercial/business schemas
    - Cobol, EDIFACT, ISO8583, NACHA
    - iCalendar, vCard

- Please publish your public-format DFDL schemas here to share with others

- US DoD NCDSMO Intelink (CAC Required)
  - FOUO/CUI Schemas
  - link16 with JREAP (MIL-STD-6016 F1)
    - some older Link16 schemas also
  - OILSTOCK
  - Demo schemas for other military formats
    - GMTIF, VMF, USMTF, …

- Please publish your FOUO/CUI schemas here for DoD shared use.

Support/Help

- Apache Daffodil project
  - join users@daffodil.apache.org mailing list
  - ask questions there - note: public archived list

Learning Resources

- github DFDLSchemas examples
- OGF DFDL v1.0 Spec (as HTML) (as PDF)
- DFDL Schema Style Guide (from Apache Daffodil project)
- Tutorials on DFDL by MITRE
- IBM
  - YouTube videos and numerous articles - search web for "IBM DFDL"
  - Getting Started with the Data Format Description Language
- Open Grid Forum DFDL Workgroup
  - Tutorials 1 - 6: Download from here

- Note: AI bots like ChatGPT and Gemini don't know much about DFDL yet. (as of December 2025)

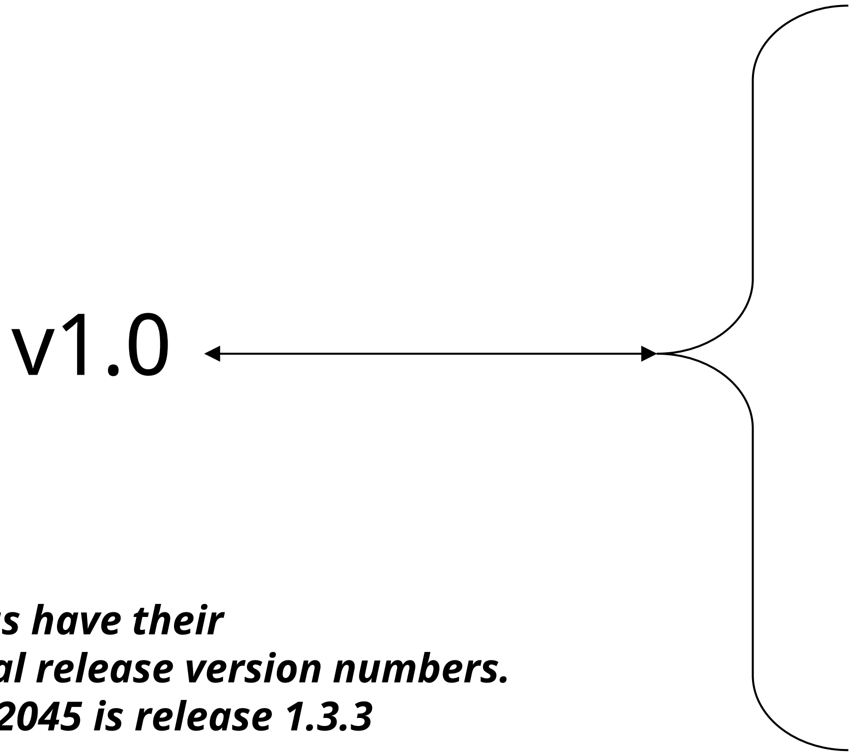
Apache Daffodil™

Status Report

Avoid Version Confusion

DFDL Language Specification

Apache Daffodil™ Software



v1.0

- **v1.0.0**
- **v1.1.0**
- **v2.0.0**
- **v2.1.0**
- **…**
- **v2.4.0**
- **…**
- **v3.0.0**
- **….**
- **v4.0.0**
- **….**

DFDL Schemas have their
own individual release version numbers.
E.g.., mil-std-2045 is release 1.3.3

Daffodil

- Open Source
  - Apache Daffodil - Announced in March 2021
  - Apache = A Permissive/Commercial Friendly License – embed how you wish, into anything.
- Runs on Java 8+ Virtual Machine
- Started as research project by UofI ~2009
- Originally developed by Owl (then Tresys) ~2012
  - Funded by the USG
  - Became Apache Incubator project in 2017
  - Graduated from Apache Incubator in March 2021
- Highly interoperable with IBM DFDL (since v2.4.0)
- Active development
- https://daffodil.apache.org

Daffodil - Recent Major Features

- EXI support
- Pluggable Character Sets
- Embedded XML Strings in data
- C-Code generator backend
- Pluggable 'layers' for Checksums, line-folding, CRC, etc.
- Schematron Support - pluggable Validators
- SAX API

Daffodil VSCode Extension

- Data Format Development and Debug IDE
  - Set data breakpoints
  - Step through parsing
  - Save TDML test cases
  - Edit large data files

- Extension to VSCode IDE
  - Front-end - Typescript
  - Back-end server - Scala

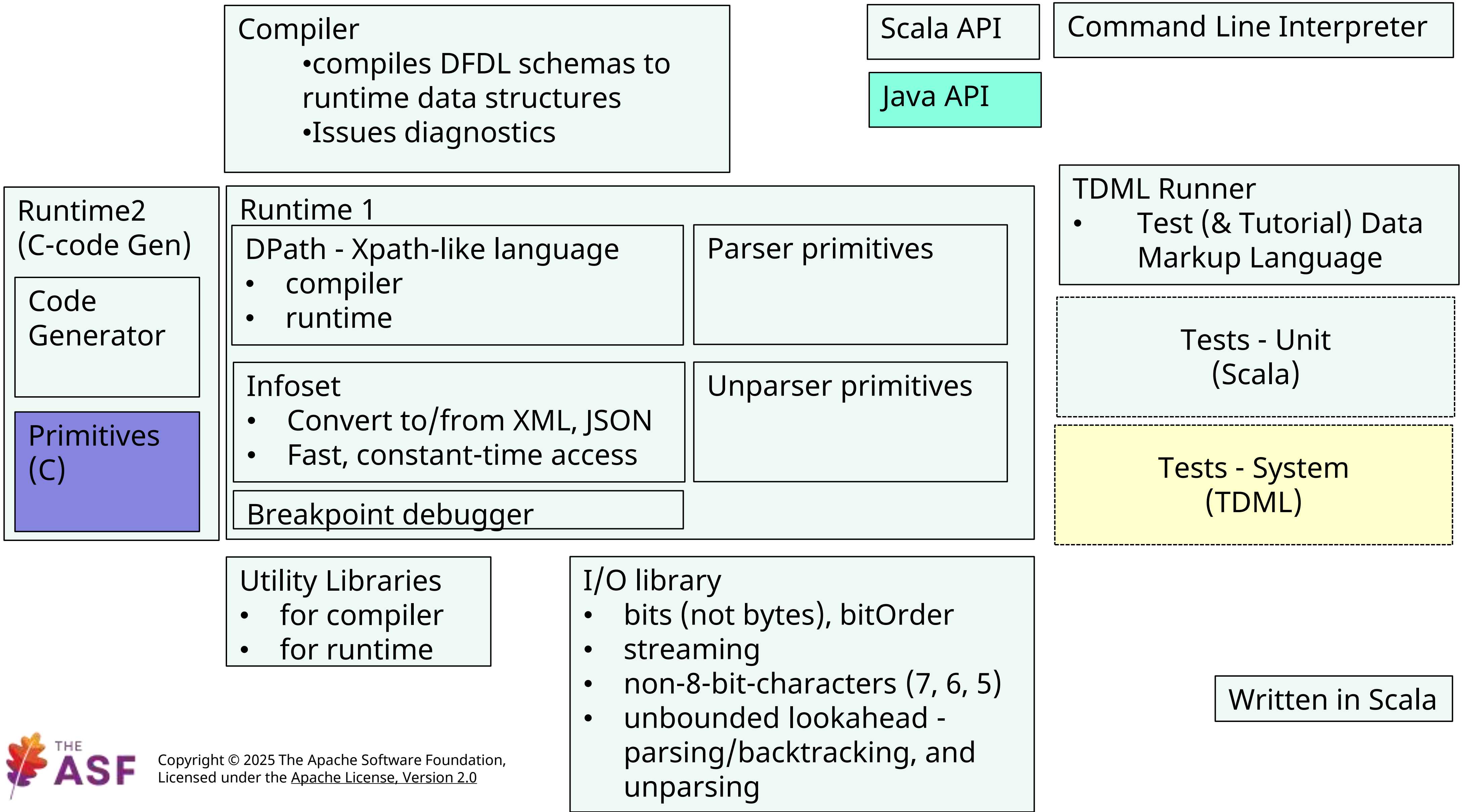
Daffodil

If you download it, what do you get?

- Jar libraries – runs on JVM
  - Compiler, utilities, TDML runner
  - Signed Jars available from Maven Central
  - Runtime 1 - Scala - runs on JVM - supports nearly all of DFDL v1.0
  - Runtime 2 - C-generator - creates fast C code - Small subset of the DFDL language
- Command Line Interface
  - Interactive CLI debugger and trace
- Java & Scala API with documentation
- XML and JSON for parse-output, unparse-input

- You must get your DFDL schemas somewhere…
  - github (DFDLSchemas, others)
  - Write them! (and share them!)

Daffodil Internal Components



Compiler
- compiles DFDL schemas to runtime data structures
- Issues diagnostics

Scala API

Java API

Command Line Interpreter

Runtime2 (C-code Gen)

Code Generator

Primitives (C)

Runtime 1

DPath - Xpath-like language
- compiler
- runtime

Parser primitives

Infoset
- Convert to/from XML, JSON
- Fast, constant-time access

Unparser primitives

Breakpoint debugger

TDML Runner
- Test (& Tutorial) Data Markup Language

Tests - Unit (Scala)

Tests - System (TDML)

Utility Libraries
- for compiler
- for runtime

I/O library
- bits (not bytes), bitOrder
- streaming
- non-8-bit-characters (7, 6, 5)
- unbounded lookahead - parsing/backtracking, and unparsing

Written in Scala

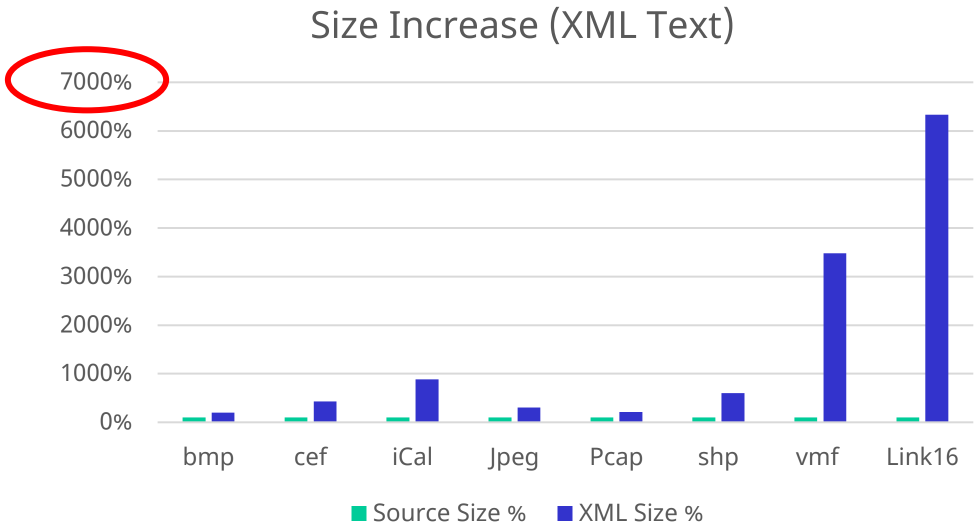
Daffodil Integrations

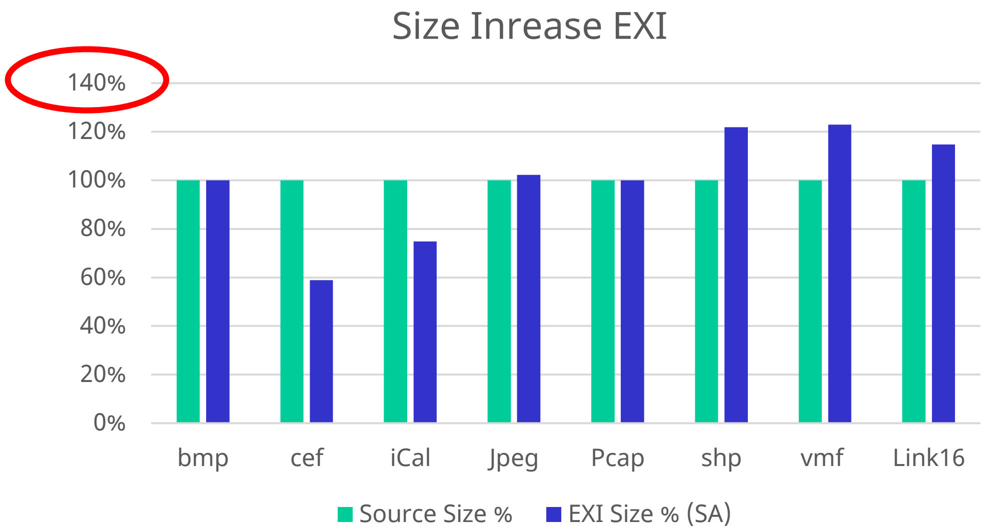
- EXI - binary form of XML - much faster/denser
- XProc - Calabash
- Apache Spark
- Apache NiFi
- Apache Drill
- SoftwareAG™ Integration Server (aka WebMethods™)
- Owl CDS products/services
- Other companies also!

- Potential
  - Apache Tika
  - Other data-handling frameworks - Flink, Hadoop,....

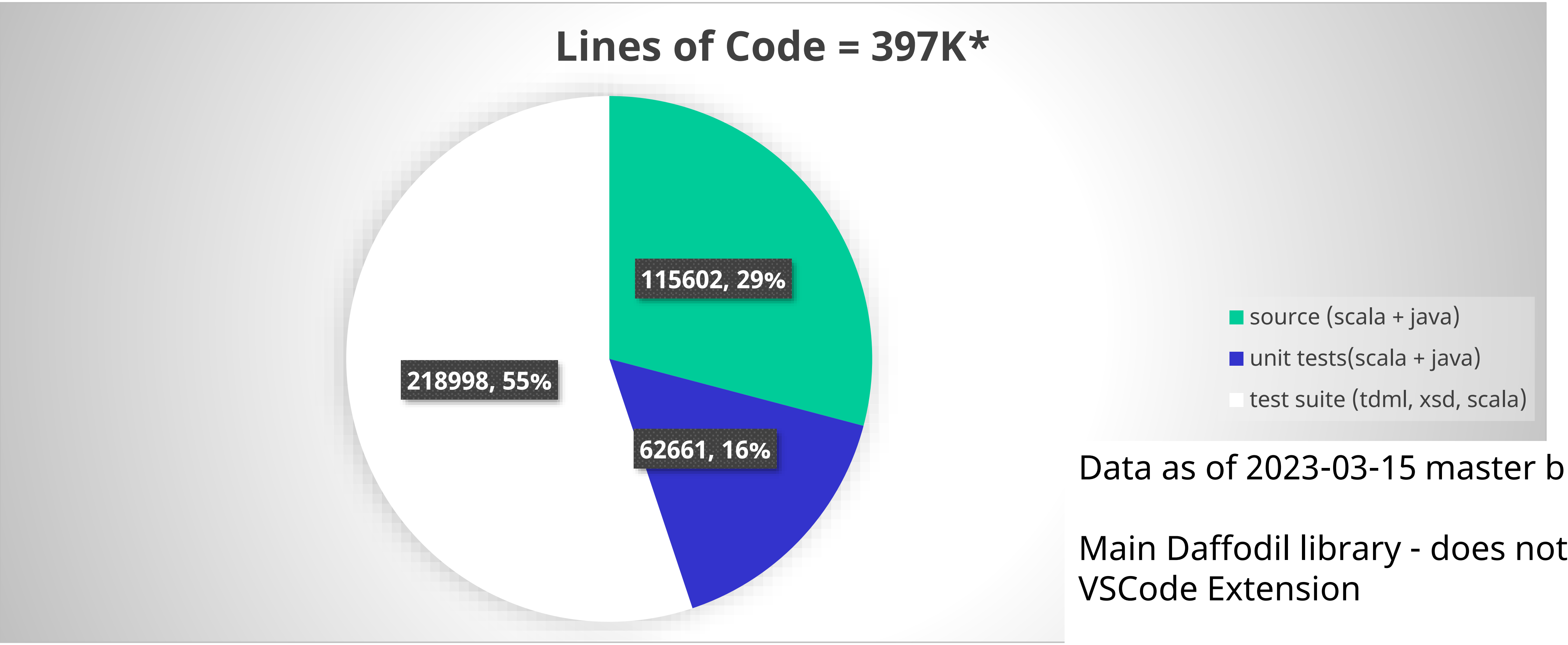
EXI (binary XML) - Data Size

- EXI is a w3c standard for a binary XML
  - Same Infoset, same API (SAX)
- Requires massively less space than textual XML
- See examples on github.com OpenDFDL





Daffodil Code Base

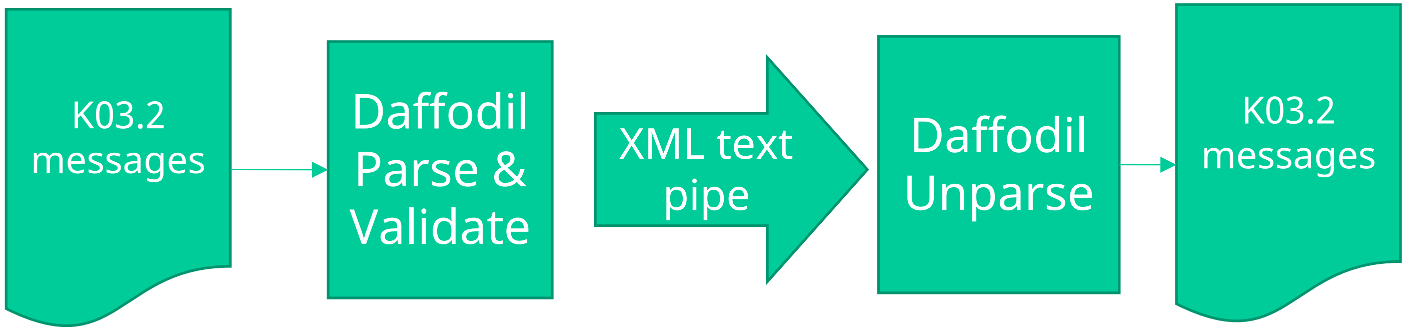


Data as of 2023-03-15 master branch

Main Daffodil library - does not include VSCode Extension

Excludes DFDL Schemas
(Link16 DFDL Schema is 800K+ LoC just by itself)

Daffodil Performance



- Uses Daffodil Message Streaming API
- The K03.2 message (from MIL-STD-6017) is 49 bytes. Expands to 4555 bytes of XML
  - Note: EXI eliminates this text expansion - see earlier slide on EXI.
  - This test was done using textual XML
- The file was 131072 repeats of that 49 byte message
- Intel Core i7-6820HQ CPU @ 2.70 Ghz (a laptop)
- Running on 2 processes (parse, unparse), 1 thread each.
- 81.218 seconds => ~ 0.6 milliseconds per message
- Conclusion:
  - DFDL/Daffodil is not going to kill your 10msec latency budget.
  - Cost will be from additional XML filtering/validation operations

Daffodil Roadmap

- Roadmap on https://s.apache.org/daffodil-wiki
- Depends on community needs, available developers

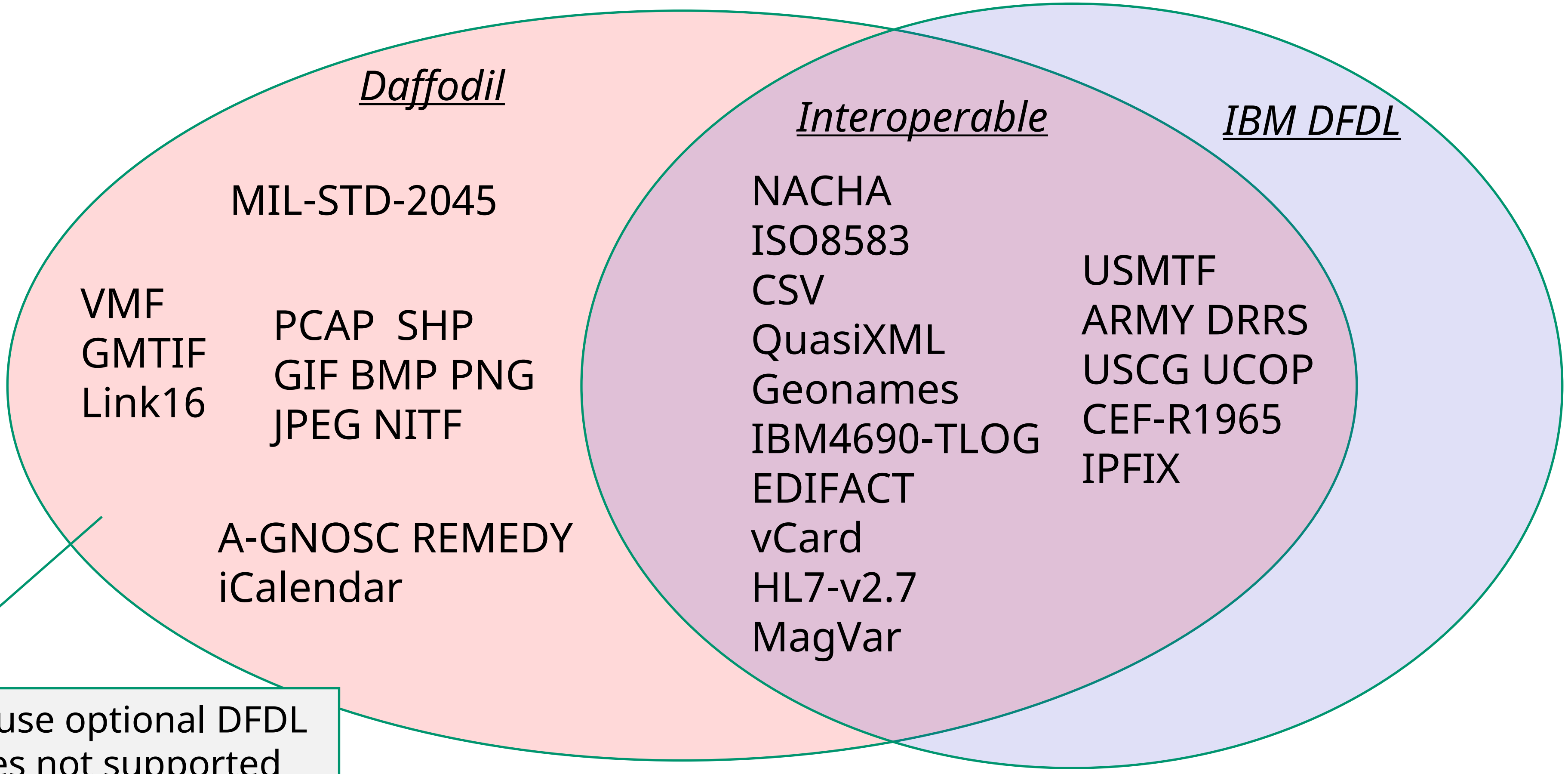
Daffodil Community

- Scala programmers wanted !
- Java programmers who want to learn Scala wanted!
- Typescript/Javascript programmers who want to work on the Daffodil VSCode Extension - wanted!

Other DFDL Implementations

- IBM DFDL - The First DFDL Implementation - v1.0 Nov 2011
  - Embeddable as Library, C and Java
  - Includes Eclipse based tooling - graphical DFDL schema editor/test environment
  - Found in IBM products:
    - IBM App Connect Enterprise product family
    - IBM Integration Bus
    - IBM InfoSphere Master Data Management
    - IBM z/TPF product family
- European Space Agency - DFDL4S = DFDL for Space
  - Embeddable as Library, Java and C++ versions.
  - Binary-data-only subset of DFDL
  - Created by ESA for satellite data descriptions
  - Evolving to be a fully compatible DFDL subset.
  - More info at
    - http://eop-cfi.esa.int/index.php/applications/dfdl4s

Interoperability – Apache Daffodil & IBM



Daffodil

MIL-STD-2045

VMF
GMTIF
Link16

PCAP SHP
GIF BMP PNG
JPEG NITF

A-GNOSC REMEDY
iCalendar

Interoperable

NACHA
ISO8583
CSV
QuasiXML
Geonames
IBM4690-TLOG
EDIFACT
vCard
HL7-v2.7
MagVar

USMTF
ARMY DRRS
USCG UCOP
CEF-R1965
IPFIX

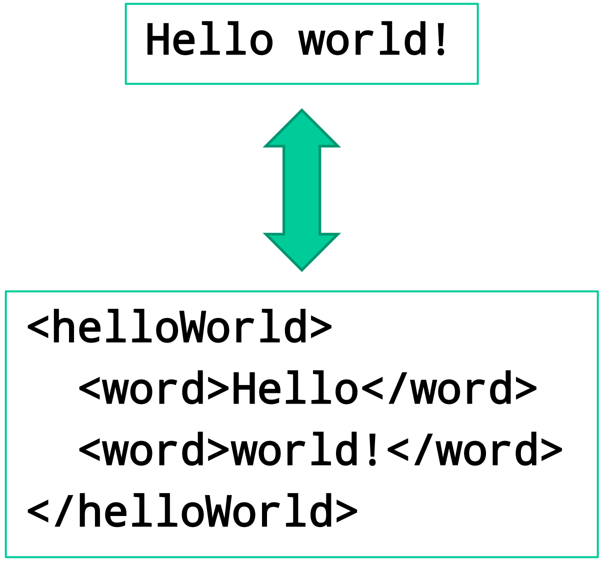
IBM DFDL

These use optional DFDL features not supported (yet) by IBM DFDL

Example: Java API

Yes, it is 'Hello world!'

This example available on github.



```
Hello world!
```

```
<helloWorld>
  <word>Hello</word>
  <word>world!</word>
</helloWorld>
```

Daffodil Java API – Hello World

```
public class HelloWorld {
  public static void main(String[] args) throws IOException {
     ... // let's fill this in
  }
}
```

```
 <!– helloWorld.dfdl.xsd -->
  …
<dfdl:format ref="daffodilTest1
   representation="text" encoding="utf-8"
   lengthKind="delimited"/>
  …
<element name="helloWorld">
  <complexType><sequence>

        <element name="word" type="xs:string"
                maxOccurs="unbounded"
                dfdl:lengthKind="delimited"
                dfdl:terminator="%WSP+;"
                dfdl:occursCountKind="parsed"/>

    </sequence></complexType>
</element>
```

Daffodil Java API – Hello World

```
//
// First compile the DFDL Schema
// Creates a Daffodil ProcessorFactory object
//
Compiler c = Daffodil.compiler();
c.setValidateDFDLSchemas(true);

File schemaFile = new File("helloWorld.dfdl.xsd");
ProcessorFactory pf = c.compileFile(schemaFile);

// list any compile-time diagnostics - warnings and errors
List<Diagnostic> diags = pf.getDiagnostics();
for (Diagnostic d : diags) {
  System.err.println(d.getSomeMessage());
}
if (pf.isError()) { System.exit(1); }
```

Daffodil Java API – Hello World

```
//
// Use ProcessorFactory to create data processors
// As many as you want. They can be run (to parse/unparse) on
// different threads.
//
DataProcessor dp = pf.onPath("/");

ReadableByteChannel rbc =
    Channels.newChannel(new FileInputStream("helloWorld.dat"));
//
// Setup JDOM outputter
//
JDOMInfosetOutputter outputter = new JDOMInfosetOutputter();

ParseResult res = dp.parse(rbc, outputter);

boolean err = res.isError();
if (err) { … list diagnostics and exit(2) … }

org.jdom2.Document doc =
   outputter.getResult(); // JDOM2 XML Infoset
```

Daffodil Java API – Hello World

```
//
// Pretty print the XML infoset.
//
XMLOutputter xo = new XMLOutputter(Format.getPrettyFormat());
xo.output(doc, System.out);

// or transform/inspect/etc.
```

Daffodil Java API – Hello World

```
//
// unparse
//

WritableByteChannel wbc =
  Channels.newChannel(new FileOutputStream(…));

JDOMInfosetInputter inputter = new JDOMInfosetInputter(doc);
UnparseResult res2 = dp.unparse(inputter, wbc);

if (res2.isError()) { … same as before … }

// File contains unparsed data
```