OGF IPR Policies Apply

- “I acknowledge that participation in this meeting is subject to the OGF Intellectual Property Policy.”
- Intellectual Property Notices Note Well: All statements related to the activities of the OGF and addressed to the OGF are subject to all provisions of Appendix B of GFD-C.1, which grants to the OGF and its participants certain licenses and rights in such statements. Such statements include verbal statements in OGF meetings, as well as written and electronic communications made at any time or place, which are addressed to:
  - the OGF plenary session,
  - any OGF working group or portion thereof,
  - the OGF Board of Directors, the GFSG, or any member thereof on behalf of the OGF,
  - the ADCOM, or any member thereof on behalf of the ADCOM,
  - any OGF mailing list, including any group list, or any other list functioning under OGF auspices,
  - the OGF Editor or the document authoring and review process
- Statements made outside of a OGF meeting, mailing list or other function, that are clearly not intended to be input to an OGF activity, group or function, are not subject to these provisions.
- Excerpt from Appendix B of GFD-C.1: “Where the OGF knows of rights, or claimed rights, the OGF secretariat shall attempt to obtain from the claimant of such rights, a written assurance that upon approval by the GFSG of the relevant OGF document(s), any party will be able to obtain the right to implement, use and distribute the technology or works when implementing, using or distributing technology based upon the specific specification(s) under openly specified, reasonable, non-discriminatory terms. The working group or research group proposing the use of the technology with respect to which the proprietary rights are claimed may assist the OGF secretariat in this effort. The results of this procedure shall not affect advancement of document, except that the GFSG may defer approval where a delay may facilitate the obtaining of such assurances. The results will, however, be recorded by the OGF Secretariat, and made available. The GFSG may also direct that a summary of the results be included in any GFD published containing the specification.”

- OGF Intellectual Property Policies are adapted from the IETF Intellectual Property Policies that support the Internet Standards Process.
Copyright (C) Open Grid Forum (2004 - 2014). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works.

The limited permissions granted above are perpetual and will not be revoked by the OGF or its successors or assignees.
Admin

- DFDL-WG co-chairs:
  - Steve Hanson, IBM UK
  - Mike Beckerle, Tresys Technology

- Two note takers please

- Sign the attendance sheet

- Note: OGF Intellectual Property Rules apply
Agenda

- DFDL Overview

- Specification Update

- Implementation: IBM DFDL

- Implementation: Daffodil

- Demonstration

- Next steps
Why DFDL?

- Grids and clouds are about universal data interchange
- Most of the world’s data is semi-structured text or binary data residing in files
- There has been no accepted standard for describing this text and binary data
  - XML -> use XML Schema
  - RDBMS -> use database schema
  - Other text/binary -> ??

*Every data handling product in the marketplace has its own proprietary way of importing/accessing data and describing data format.*

- Existing standards are not flexible enough
  - Prescriptive: “Put your data in this format!”
  - Examples: ASN.1, XDR, GPB, Thrift, Avro, …
  - You must use one of the defined encodings & syntax

✓ **DFDL**: a universal, shareable, non-prescriptive, description for general text & binary data formats
Data Format Description Language (DFDL)

• An **open** standard from OGF
  – Version 1.0
  – ‘Proposed Recommendation’ status

• A way of **describing** data…
  – It is NOT a data format itself!

• A **powerful** modeling language …
  – Text, binary and bit
  – Commercial record-oriented
  – Scientific and numeric
  – Modern and legacy
  – Industry standards

• While allowing **high performance** …
  – You choose the right data format for the job

• Leverage **XML Schema** technology
  – Uses W3C XML Schema 1.0 subset & type system to describe the logical structure of the data
  – Uses XSDL annotations to describe the physical representation of the data
  – The result is a **DFDL schema**

• Keep simple cases **simple**
• Annotations are **human readable**
• Both **read and write**
  – A **DFDL processor** can parse and serialize data in described format from same DFDL schema

• **Intelligent** parsing
  – Automatically resolve choice and optionality

• **Validation** of data when parsing and serializing
Example – Delimited Text Data

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

```xml
<xs:complexType name="numbers">
  <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="category" type="xs:int">
      <xs:annotation>
        <xs:appinfo source="http://www.ogf.org/dfdl/v1.0">
          <dfdl:element representation="text" textNumberPattern="#0" encoding="ascii" lengthKind="delimited" initiator="cat=" ... />
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
    <xs:element name="lowerBound" type="xs:float">
      <xs:annotation>
        <xs:appinfo source="http://www.ogf.org/dfdl/v1.0">
          <dfdl:element representation="text" textNumberPattern="#0.0#E0" encoding="ascii" lengthKind="delimited" initiator="lbound=" ... />
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

**DFDL annotation**

**DFDL properties**

```xml
cat=5; lbound=-7.1E8
```
Example – DFDL schema (short form)

```
<xs:complexType name="numbers">
  <xs:sequence dfdl:separator=";" dfdl:encoding="ascii" ... >
    <xs:element name="category" type="xs:int"
      dfdl:representation="text"
      dfdl:textNumberPattern="#0" dfdl:encoding="ascii"
      dfdl:lengthKind="delimited" dfdl:initiator="cat=" ... />
    <xs:element name="lowerBound" type="xs:float"
      dfdl:representation="text"
      dfdl:textNumberPattern="#0.0#E0" dfdl:encoding="ascii"
      dfdl:lengthKind="delimited" dfdl:initiator="lbound=" ... />
  </xs:sequence>
</xs:complexType>
```

DFDL properties
DFDL Processor

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

```
<xs:complexType name="numbers">
  <xs:sequence dfdl:separator=";" dfdl:encoding="ascii" ... >
    <xs:element name="category" type="xs:int"
        dfdl:representation="text"
        dfdl:encoding="ascii" dfdl:textNumberPattern="###0"
        dfdl:lengthKind="delimited" dfdl:initiator="cat=" ... />
    <xs:element name="lowerBound" type="xs:float"
        dfdl:representation="text"
        dfdl:encoding="ascii" dfdl:textNumberPattern="##0.0#E0"
        dfdl:lengthKind="delimited" dfdl:initiator="lbound=" ... />
  </xs:sequence>
</xs:complexType>
```
DFDL 1.0 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
- 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
When should I use DFDL?

- DFDL’s sweet spot is when you need to model and parse a text or binary data format and where either:
  - You have a specification of the data format ‘on the wire’
  - You have actual wire examples of the data format

- DFDL is recommended to model:
  - Binary data from COBOL, C, PL/1, ASM programs
  - Text data with delimiters such as CSV
  - Text industry standards such as SWIFT, HL7, EDIFACT, X12, …
  - Binary industry standards such as ISO8583, Tlog, PCAP, …

- DFDL is not recommended to model:
  - XML
    - Already have XML parsers and XML Schema / DTDs
  - JSON
    - Already have JSON parsers, and JSON schema under design
  - GPB, HDF5, …
    - With serialization formats like GPB, the wire format is never exposed to the consumer and access to the data is using APIs
Agenda

- DFDL Overview
  - Specification Update
    - Implementation: IBM DFDL
    - Implementation: Daffodil
  - Demonstration
Status

- DFDL 1.0 spec is currently a *Proposed Recommendation*
  - GFD.174 P-REC published Feb 2011
  - GFD.207 P-REC published Nov 2014 (obsoletes GFD.174)

- Progress towards *Recommendation*
  - Two implementations underway – IBM® DFDL & Daffodil
  - Implementations conform to GFD.207

- Supplementary OGF publications
  - GFD.190 INFO Mapping between DFDL 1.0 Infoset and XDM
  - GFD.197 INFO Example set of DFDL 1.0 properties
  - GFD.214 EXP #1 Errata for DFDL 1.0 Specification GFD.174
  - GFD.215 EXP #2 Empty, Missing, Defaults, Arrays
  - GWD.xxx EXP #3 Bit Order

- Any errata to GFD.207 are tracked as Redmine issues
Conformance

- DFDL 1.0 specification is not small! (over 200 pages)
- DFDL-WG want to make it easier to create conforming processors
- Conformance can be claimed separately for DFDL Parser & Unparser
- The features of DFDL are divided into Core and Optional
- A DFDL Parser, Unparser or Processor can claim to be:
  - Minimal
  - Extended
  - Full
- Example: A Minimal DFDL Parser implements just the parser and all Core features
- Example: An Extended DFDL Processor implements both parser and unparser, all Core features plus some Optional features
- Conformance test suite desirable (major undertaking)
Easier, non-normative way to learn DFDL
  – Same idea as XML Schema 1.0 Primer

Divided into example-based Lessons so you can learn at your own pace
  1. Introduction
  2. Language Basics
  3. DFDL Properties
  4. Modeling Basic Structures
  5. Modeling Alternative Structures
  6. Modeling Optional and Repeating Data
  7. ...

Lessons 1 to 6 available on Redmine
Web Community for DFDL Schemas

- Free public repository for DFDL models
- Hosted on GitHub
- Unlimited read-only access
- Evolving content
- Industry standard formats
- Collaboration encouraged
Getting Started with DFDL

Get started with the Data Format Description Language
An open standard for data modeling

In this age of big data, the bulk of the data begging to be analyzed is not XML, but rather it is other structured and semi-structured formats, both text and binary. Until now, no open standard has been developed that is capable of describing a wide variety of such data formats. Learn about the Open Grid Forum (OGF) proposed recommendation for a powerful language that describes many different data formats, the Data Format Description Language (DFDL).

Introduction to DFDL

In this first video, Steve Hanson introduces you to the motivations and design goals for DFDL and shows you some of the key features of the specification. He also describes the features of the IBM DFDL component and discusses the open source implementation of DFDL called Daffodil. Finally, he gives you a quick guided tour of the IBM DFDL component in an example with IBM Integration Bus.

IBM Integration Bus for Developers
See the IBM DFDL component in action by downloading the "IBM Integration Bus for Developers."

Agenda

- DFDL Overview
- Specification Update
  - Implementation: IBM DFDL
  - Implementation: Daffodil
- Demonstration
- Designed as an embeddable component for IBM products including:
  - IBM Integration Bus v9
  - Rational® Test Workbench v8.0.1
  - InfoSphere® Master Data Management v11
  - Other IBM products will adopt

- DFDL processor
  - High performance parser and unparserr
  - Java and C
  - Pre-compiles DFDL schema for performance
  - Parser emits SAX-like events
  - Streaming, on-demand, speculative
  - Compliance test suite (17000+) using .tdml files

- Tools for creating DFDL schema
  - Graphical DFDL schema editor
  - Wizards for CSV, COBOL & C
  - Graphical DFDL schema tester
  - Shipped as eclipse plugins
IBM DFDL - Status

- IBM DFDL implements about 85% of the OGF DFDL 1.0 specification
  - Support for rest will be added in future IBM DFDL releases

- Currently unsupported:
  - Bi-directional text
  - Floating elements
  - Hidden groups
  - Calculated values
  - Certain XPath functions
  - Arrays with stop values
  - Default values (parser)

- IBM DFDL implements many of the identified spec errata

- In production use by many IBM customers!

- Can I use it?
  - Yes. IBM DFDL Java parser & unparsers can be unbundled from IBM Integration Bus and used in stand-alone Java applications
  - Free developer edition available: https://ibm.biz/iibopenbeta
IBM DFDL Tools – Schema Editor

Logical structure view

DFDL properties view
IBM DFDL Tools – Schema Tester

Run parser

Parsed 'infoset'

Parsed data

Delimiters highlighted

Hex view
IBM DFDL Tools – Schema Tester

Object in error

Parsed 'infoset' up to error

Error message

Parsed data up to error

Trace console
Agenda

- DFDL Overview
- Specification Update
- Implementation: IBM DFDL
  - Implementation: Daffodil
- Demonstration
Open source project hosted at University of Illinois
- Uses Scala programming language
  - Java compatible. Runs on JVM. Has Java-callable API
  - Univ of Illinois license (very BSD-like)

Goal is to implement the entire DFDL specification
- Parser (first) and unparsers (second)
- DOM-tree-style implementation (now)
- Streaming/event & random-access (future)
- Compliance test suite (1800 tests and growing)

Active contributors from both corporations and US government labs
- ~3 core funded software developers
- ~3 core funded test engineers

Details are available on the project Wiki
- https://opensource.ncsa.illinois.edu/confluence/display/DFDL

Contributors and Users are welcome!
Daffodil - Status

- **History**
  - Started at the Univ of Illinois' National Center for Supercomputing Applications
    - With funding from US National Archives and Records Administration
  - Designed for an earlier working-copy of the DFDL specification

- **Activity**
  - Code-base has been rewritten to conform to current DFDL 1.0 specification
  - New compiler-style front-end that parses DFDL schemas and constructs a robust abstract syntax tree of the schema

- **Themes**
  - Design-for-Test (DFT) - conformance tests means lots of tests
    - Re-using the .tdml test format from IBM DFDL
  - A declarative and functional coding style

- **Can I use it?**
  - Yes. Initial release April 2013. Approximately monthly ‘spins’ will update.
  - [https://opensource.ncsa.illinois.edu/confluence/display/DFDL/Getting+Daffodil](https://opensource.ncsa.illinois.edu/confluence/display/DFDL/Getting+Daffodil)
Daffodil Tools – Command Line Interface

$ daffodil --help

Usage: daffodil [GLOBAL_OPTS] <subcommand> [SUBCOMMAND_OPTS]

Global Options:
   -d, --debug [file]  enable debugging. Optionally, read initial debugger commands from [file] if provided.
   -t, --trace         run the debugger with verbose trace output
   -v, --verbose       increment verbosity level, one level for each -v (default = 0)
   --help              Show help message
   --version           Show version of this program

Subcommands:
   parse      parse data to a DFDL infoset
   test       list or execute TDML tests

Run 'daffodil <subcommand> --help' for subcommand specific options
Daffodil Tools – Command Line Interface

```
$ # Let's parse some PCAP data
$
$ daffodil parse -s examples/pcap/pcap.dfdl.xsd examples/pcap/icmp.cap

<pCap:PCAP xmlns:pcap="urn:pcap:2.4">
  <pcap:PCAPHeader>
    <pcap:MagicNumber>D4C3B2A1</pcap:MagicNumber>
    <pcap:Version>
      <pcap:Major>2</pcap:Major>
      <pcap:Minor>4</pcap:Minor>
    </pcap:Version>
    <pcap:Zone>0</pcap:Zone>
    <pcap:SigFigs>0</pcap:SigFigs>
    ... Many more lines of XML here ...
  </pcap:PCAPHeader>
</pcap:PCAP>
$```
Daffodil Tools – Command Line Interface

```bash
$ # Let’s run a TDML (i.e., self contained) test file
$ # TDML is best way to create small tests, report bugs, etc.
$
$ daffodil test examples/pcap/pcap.tdml

[Pass] pcap_test_dns
[Pass] pcap_test_http_ipv6
[Pass] pcap_test_icmp
[Pass] pcap_test_tcp_ecn

Total: 4, Pass: 4, Fail: 0, Not Found: 0
$
```
Daffodil Tools – Interactive Debugger

```bash
$ # Let's do some single step debugging
$

$ daffodil -d parse -s examples/pcap/pcap.dfdl.xsd examples/pcap/icmp.cap

(debug) help

  break           create a breakpoint
  clear           clear the screen
  complete        disable all debugger actions and continue
  condition       set a DFDL expression to stop at breakpoint
  continue        continue parsing until a breakpoint is found
  delete          delete breakpoints and displays
  disable         disable breakpoints and displays
  display         show value of expression each time program stops
  enable          enable breakpoints and displays
  eval            evaluate a DFDL expression
  help            display information about a command
  history         display the history of commands
  info            display information
  quit            immediately abort all processing
  set             modify debugger configuration
  step            execute a single parser step
  trace           same as continue, but runs display commands during every step

(debug)
```
... Bunch of steps ...
(debug) display eval .
(debug) display info diff
(debug) display info data
(debug) step
step
<pcap:Version xmlns:pcap="urn:pcap:2.4">
    <pcap:Major>2</pcap:Major>
</pcap:Version>
diff:
    position (bytes): 4 -> 6
data:

0xD4C3B2A1020004000000000000000000FFFF000001000000C46FC151F8CC0C004A0000004A0000000506E01449000C29340BDE08004500003CD74300080012B73C0A89E8B
(debug)
Daffodil Tools – Interactive Debugger

Can evaluate expressions and “look around” at the data resulting from the parse.

```scheme
(d debug) eval ../..
<pcap:PCAPHeader xmlns:pcap="urn:pcap:2.4">
  <pcap:MagicNumber>D4C3B2A1</pcap:MagicNumber>
  <pcap:Version>
    <pcap:Major/>
  </pcap:Version>
</pcap:PCAPHeader>
(d debug) eval ../../pcap:MagicNumber
<pcap:MagicNumber xmlns:pcap="urn:pcap:2.4">D4C3B2A1</pcap:MagicNumber>
(d debug)
```
Agenda

- DFDL Language Overview
- Specification Update
- Implementation: IBM DFDL 1.0
- Implementation: Open source Daffodil
- Demonstration
Links

- IBM DFDL via IBM Integration Bus: [https://ibm.biz/iibopenbeta](https://ibm.biz/iibopenbeta)
- Open Source Daffodil: [http://opensource.ncsa.illinois.edu/confluence/display/DFDL](http://opensource.ncsa.illinois.edu/confluence/display/DFDL)
- DFDL Schemas on GitHub: [https://github.com/DFDLSchemas](https://github.com/DFDLSchemas)
Extra Slides Follow this Marker Slide

The slides past this marker may be of value to some audiences and so are kept here for situational use by the presenter.
DFDL Evolved from….

- **Products/Technologies**
  - Mercator, Ascential, Torrent, IBM Message Broker, OMG CORBA, Microsoft BizTalk, SAS, and others.
  - Database loaders
  - COBOL/Legacy

- **Data formats:**
  - SWIFT, HL7, EDIFACT, FIX, X12, ISO8583(CCards), ASN.1 PER, Thomson Financial, .... many others