OpenGridForum
OPEN FORUM | OPEN STANDARDS

DFDL 1.0 – Specification and Implementation Update

Steve Hanson, smh@uk.ibm.com
Mike Beckerle, mbeckerle@tresys.com

April 2013

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.

Full Copyright Notice

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

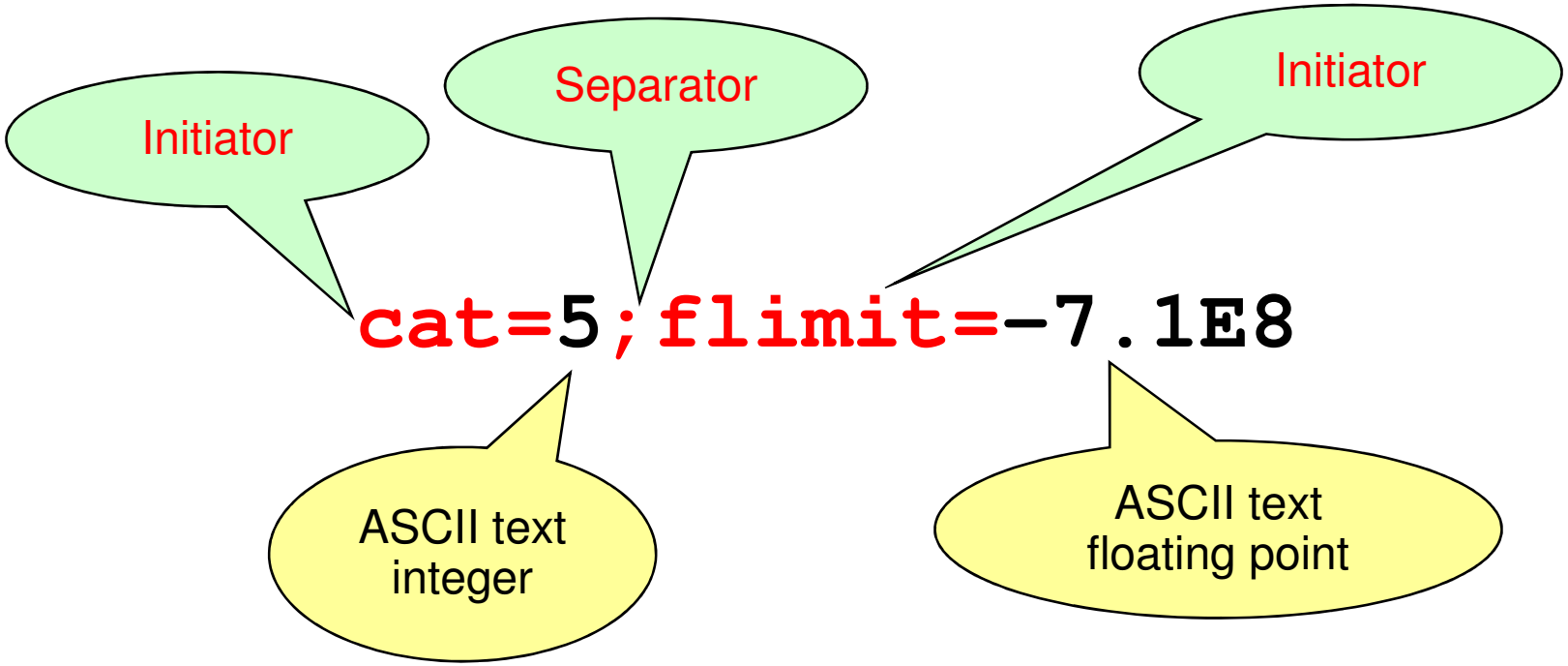
✓ ***DFDL:*** *a universal, shareable, non-prescriptive, description for general text & binary data formats*

Data Format Description Language (DFDL)

- A new ***open*** standard
  - 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***
- Both ***read and write***
  - Parse and serialize data in described format from same DFDL schema
- Keep simple cases ***simple***
  - Annotations are ***human readable***
- ***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

```
<xs:complexType name="myNumbers">
  <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="flimit" 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="flimit=" .../>
       </xs:appinfo>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

;

cat=5

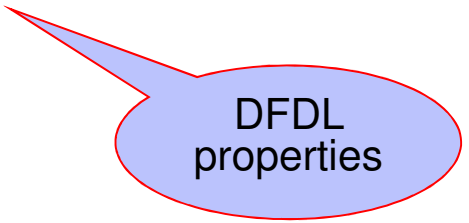
flimit=
-7.1E8

DFDL annotation

DFDL properties

Example – DFDL schema (short form)

```
<xs:complexType name="myNumbers">
  <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="flimit" type="xs:float"
                dfdl:representation="text"
                dfdl:textNumberPattern="#0.0#E0" dfdl:encoding="ascii"
                dfdl:lengthKind="delimited" dfdl:initiator="flimit=" ... />
</xs:sequence>
</xs:complexType>
```

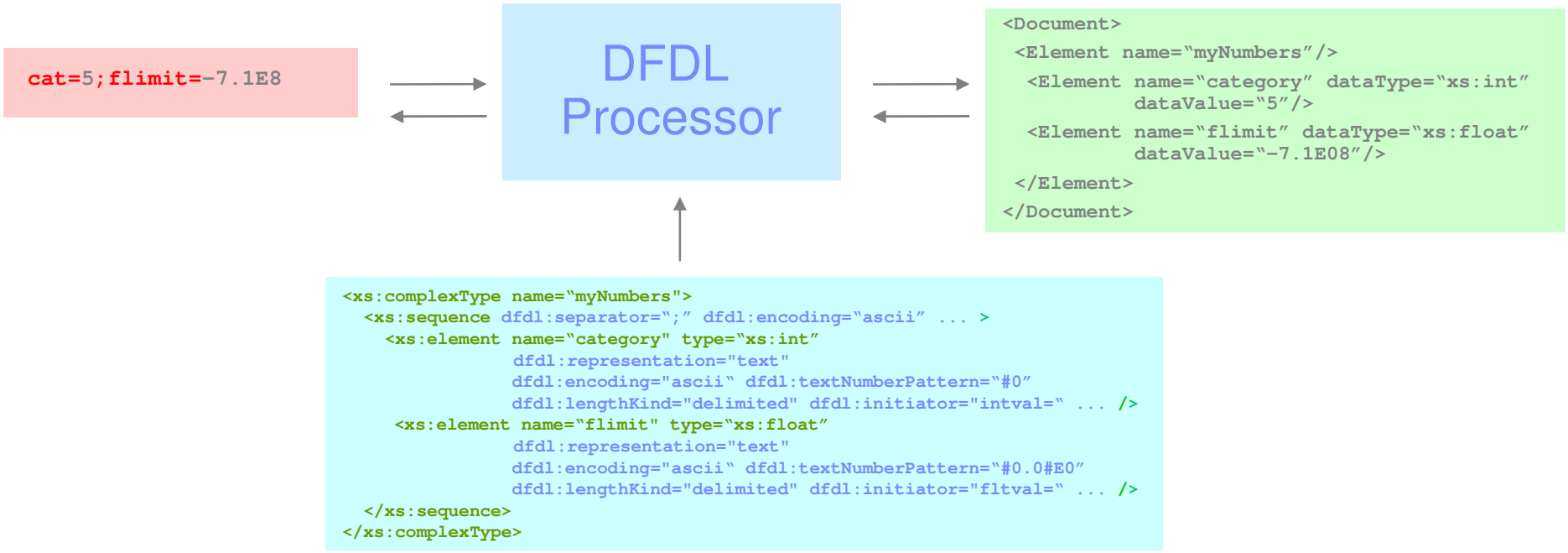


DFDL features

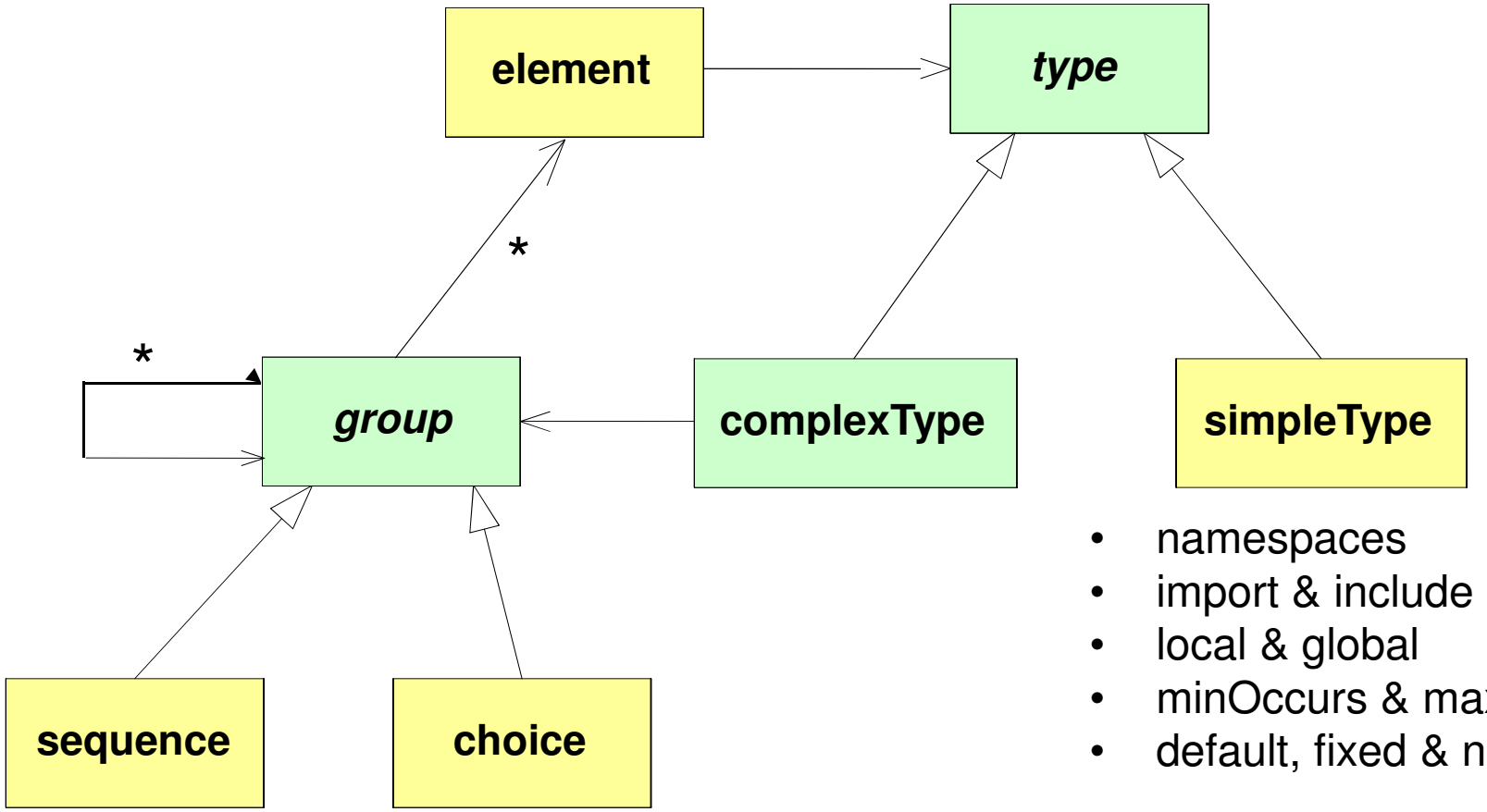
- Text data types such as strings, numbers, zoned decimals, calendars, booleans
- Binary data types such as integers, floats, BCD, packed decimals, calendars, booleans
- Fixed length data and data delimited by text or binary markup
- Language data structures found in COBOL, C and PL/1
- Industry standards such as SWIFT, HL7, FIX, HIPAA, X12, EDIFACT, ISO8583
- 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

DFDL Processor

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



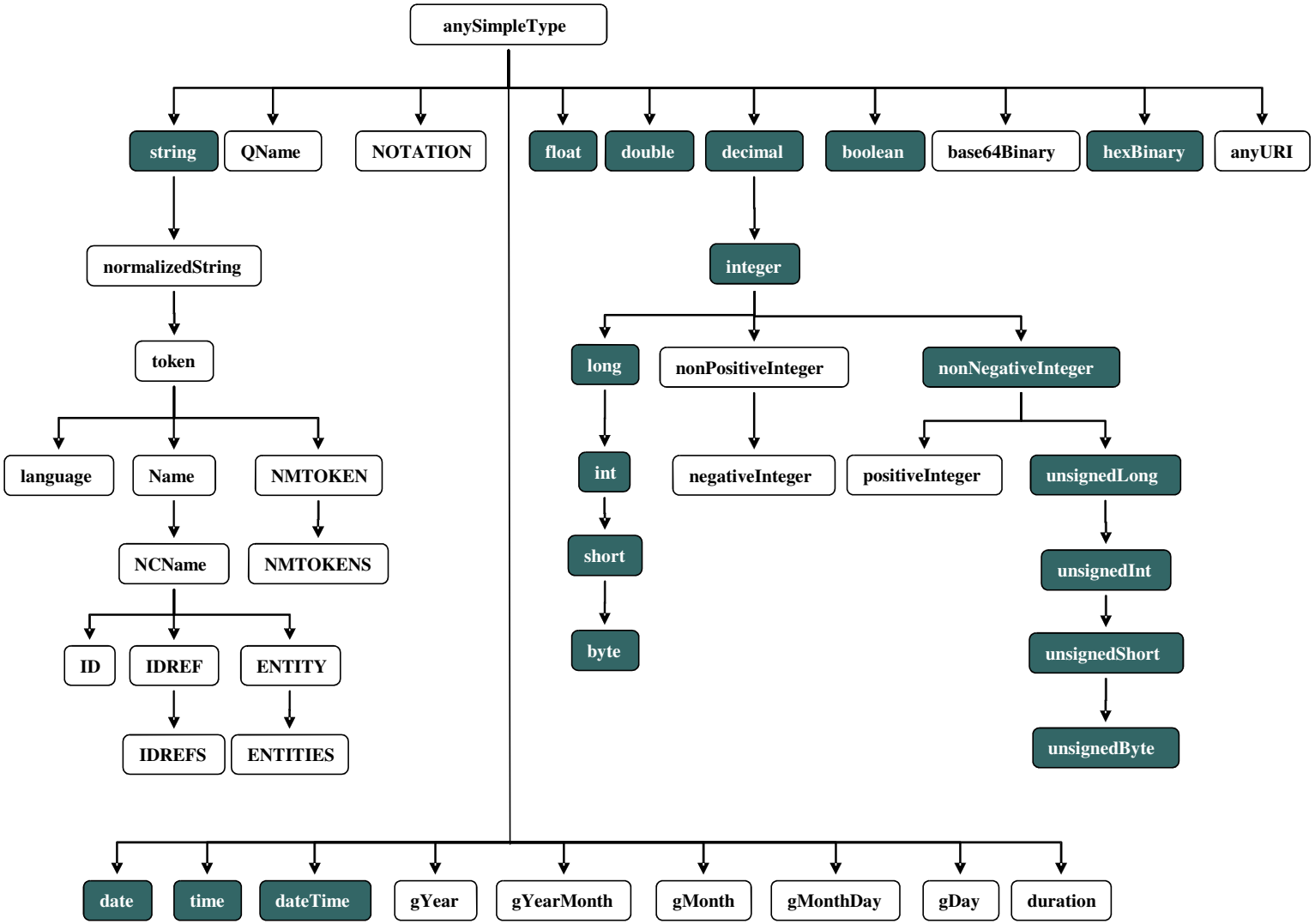
XML Schema subset



- namespaces
- import & include
- local & global
- minOccurs & maxOccurs
- default, fixed & nillable

DFDL properties are placed on yellow objects only

Supported Simple Types



DFDL language – basic annotations

Annotation	*Used on Component*	*Purpose*
dfdl:element	xs:element xs:element reference	Contains the DFDL properties of an xs:element and xs:element reference
dfdl:choice	xs:choice	Contains the DFDL properties of an xs:choice.
dfdl:sequence	xs:sequence	Contains the DFDL properties of an xs:sequence.
dfdl:group	xs:group reference	Contains the DFDL properties of an xs:group reference to a group definition containing an xs:sequence or xs:choice.
dfdl:simpleType	xs:simpleType	Contains the DFDL properties of an xs:simpleType
dfdl:format	xs:schema dfdl:defineFormat	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.
dfdl:defineFormat	xs:schema	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.

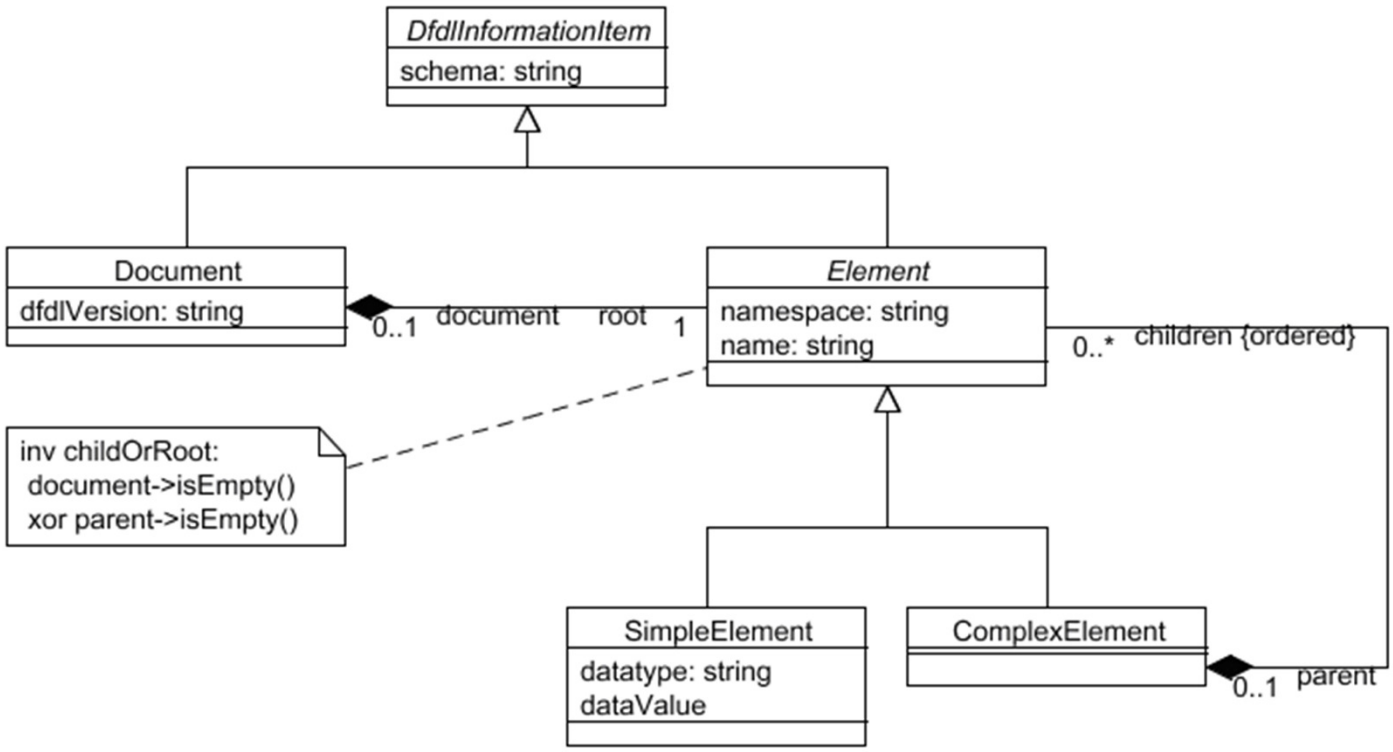
DFDL language – advanced annotations

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

DFDL Information Set

- An abstract data set defining the content that must be provided
  - To an application by a DFDL Parser
  - To a DFDL Unparser by an application

- Same concept as XML Data Model (XDM)



Speculative parsing

- 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'*

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 in the schema's special dfdl:format annotation, where it acts as a default for all objects in the schema
3. 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

- 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

aaa;bbb@;ccc;ddd%;

```
<xs:schema>
  <xs:annotation>
   <xs:appinfo source="http://www.ogf.org/dfdl/" >
      <dfdl:format terminator=";" … />
   </xs:appinfo>
  </xs:annotation>

  <xs:annotation>
   <xs:appinfo source="http://www.ogf.org/dfdl/" >
      <dfdl:defineFormat name="b_style" />
         <dfdl:format terminator="@;" … />
      </dfdl:defineFormat>
   </xs:appinfo>
  </xs:annotation>

  <xs:complexType>
    <xs:sequence dfdl:terminator="">
      <xs:element name="a" type="xs:string" />
      <xs:element name="b" type="xs:string" dfdl:ref="b_style" />
      <xs:element name="c" type="xs:string" />
      <xs:element name="d" type="xs:string" dfdl:terminator="%;"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Default field terminator is ";" but can vary

Property from schema's dfdl:format

Property via dfdl:ref

Property set locally

DFDL Expressions

- DFDL provides an expression language that can be used at various places in a DFDL schema:
  - When a property value needs to be set dynamically from the contents of the data
  - In an assert or discriminator annotation
  - When setting the value or default value of a variable
- The expression language is a subset of XPath 2.0, including variables, and with some extra DFDL-specific functions
- Expressions are always enclosed by curly braces { }

```
<xs:complexType>
  <xs:sequence dfdl:separator=“,” ... >
    <xs:element name=”count” type=”xs:nonNegativeInteger”
                dfdl:representation=“text” dfdl:lengthKind=“delimited”
                dfdl:textNumberPattern=“#0” ... />
    <xs:element name=”value” type=”xs:string” maxOccurs=“unbounded”
                dfdl:lengthKind=“delimited”
                dfdl:occursCountKind=“expression”
                dfdl:occursCount=“{../count}” ... />
  </xs:sequence>
</xs:complexType>
```

Agenda

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

Status

- DFDL 1.0 spec is currently a *Proposed Recommendation*
  - GFD-P-R.174
  - Since Feb 2011
- Progress towards *Recommendation*
  - Two implementations underway
  - Several errata found
  - New draft (imminent) will require another *Public Comment* phase
- Supplementary publications
  - HTML version of spec (courtesy IBM)
  - Multi-part tutorial (in progress)
  - GFD-I.190 Mapping between DFDL Infoset and XDM
  - GWD-I Example set of DFDL properties (in *Public Comment*)

Errata

- Errata are being tracked in a separate DFDL-WG document on Redmine
- Divided into Editorial, Minor and Major
- Minor errata are typically clarifications to behaviour
- Major errata imply additional or corrected behaviour
- Examples of Major errata:
  - New failure type ‘recoverable error’ for use by dfdl:assert
  - Enhance Infoset to enable W3C PSVI to be built
  - Revise handling of Unicode BOMs
  - Limit dfdl:lengthKind ‘pattern’ to text data
  - Allow complex elements to be ‘nillable’
  - Semantic of validation when unparsing
  - Control of encoding/decoding errors
  - Renaming dfdl:separatorPolicy
  - Additional dfdl:occursCountKind ‘implicit’
  - Clarification of ‘empty’ and ‘missing’ and when to apply default values

Conformance

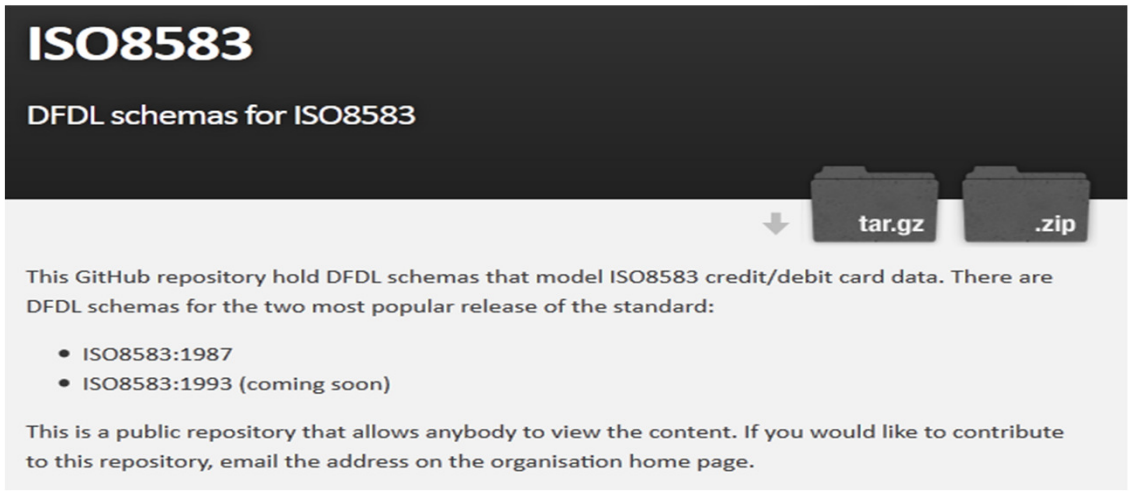
- DFDL 1.0 specification is not small! (nearly 200 pages)
- DFDL-WG want to make it easier to create conforming processors
- Conformance can be claimed separately for DFDL Parser & Unparser
- The features of DFDL are divided into Core and Optional
- A DFDL Parser, Unparser or Processor can claim to be:
  - 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)

Tutorial

- 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. Modeling Text Values
  8. Modeling Binary Values
  9. …
- Drafts available for Lessons 1 to 6

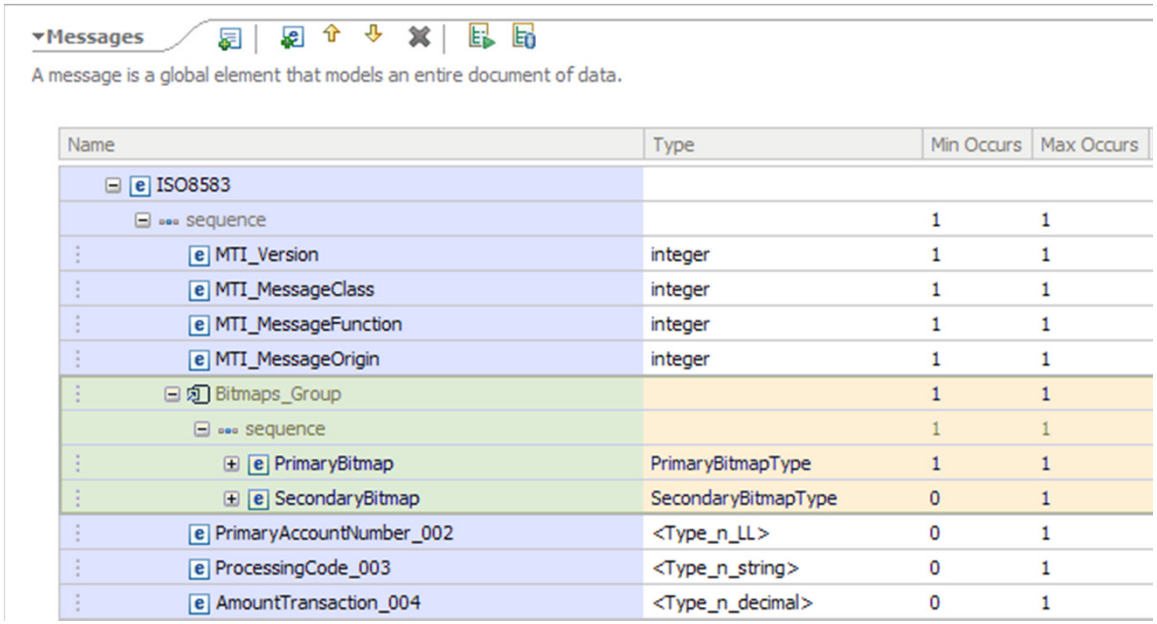
Web Community for DFDL Schemas

- Free public repository for DFDL models
- Hosted on the popular GitHub community website
- Unlimited read-only access
- Collaboration encouraged
- Evolving content



Industry Formats

- DFDL schemas are emerging for well-known data formats:
- HL7 v2.5.1, v2.6 and v2.7
  - From IBM WebSphere Message Broker Connectivity Pack for Healthcare
- IBM/Toshiba 4690 SurePos ACE v7r3 TLOG
  - DFDLSchemas on GitHub
- ISO 8583 (1987)
  - DFDLSchemas on GitHub
  - IBM WebSphere Message Broker sample
- More to follow …



Agenda

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

IBM DFDL 1.0

- Designed as an embeddable component
  - Shipped with various IBM products such as:
    - WebSphere Message Broker v8
    - Rational Test Workbench v8.0.1
  - Other IBM products and appliances will adopt

- DFDL processor
  - High performance Parser and Unparser
  - Java and C versions
  - Uses compiled schema (grammar)
  - SAX-like events
  - Streaming, on-demand, speculative
  - Compliance tests (14000+)
- Tooling for creating DFDL models
  - Eclipse plugins
  - Guided authoring wizards
  - COBOL importer wizard
  - DFDL schema editor (see demo)
  - Debug model using real data from within tooling (see demo)

IBM DFDL 1.0 - Status

- IBM DFDL 1.0 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
  - Unordered groups & floating elements
  - Regular expressions (coming soon)
  - Hidden groups & calculated values
  - Certain XPath functions
  - Arrays with stop values
  - Default values (parser)
- IBM DFDL 1.0 implements majority of the identified spec errata
- In production use by IBM customers!
- IBM DFDL Java parser & unparser can be unbundled from WebSphere Message Broker and used in stand-alone applications
- Free developer edition available

Agenda

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

Open Source Daffodil

- Hosted at University of Illinois
  - Uses *Scala* programming language
  - Univ of Illinois license (very BSD-like)

- Goal is to implement the entire DFDL specification
  - Parser (first) and Unparser (second)
  - DOM-tree-style implementation (now)
  - Streaming/event & random-access (future)
  - Compliance test suite (1000 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
  - Contributors are welcome!

Open Source 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 provide conformance 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.

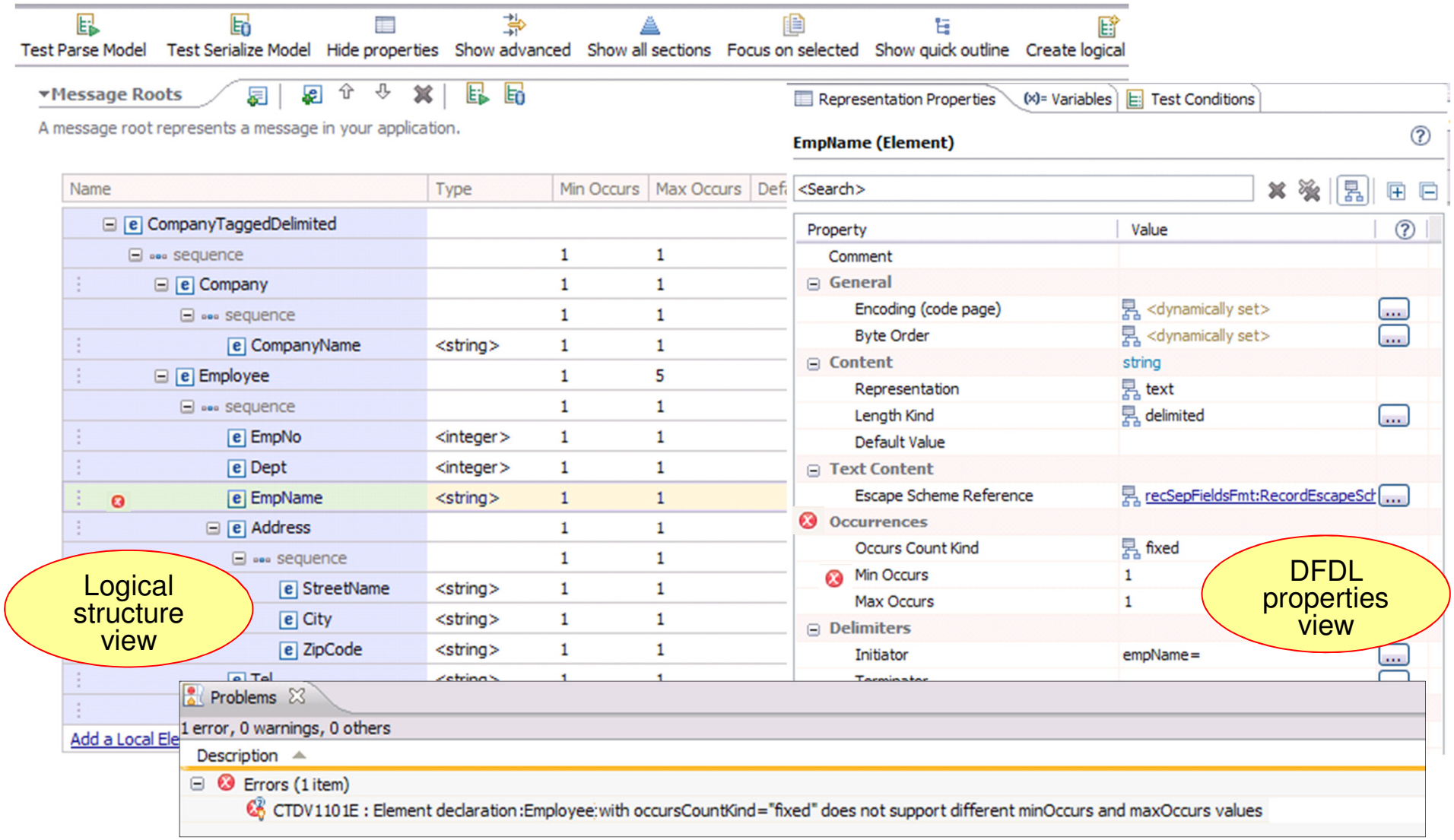
Daffodil Tools

- Screen shots here of the command line tool
- Screen shots of the interactive debugger

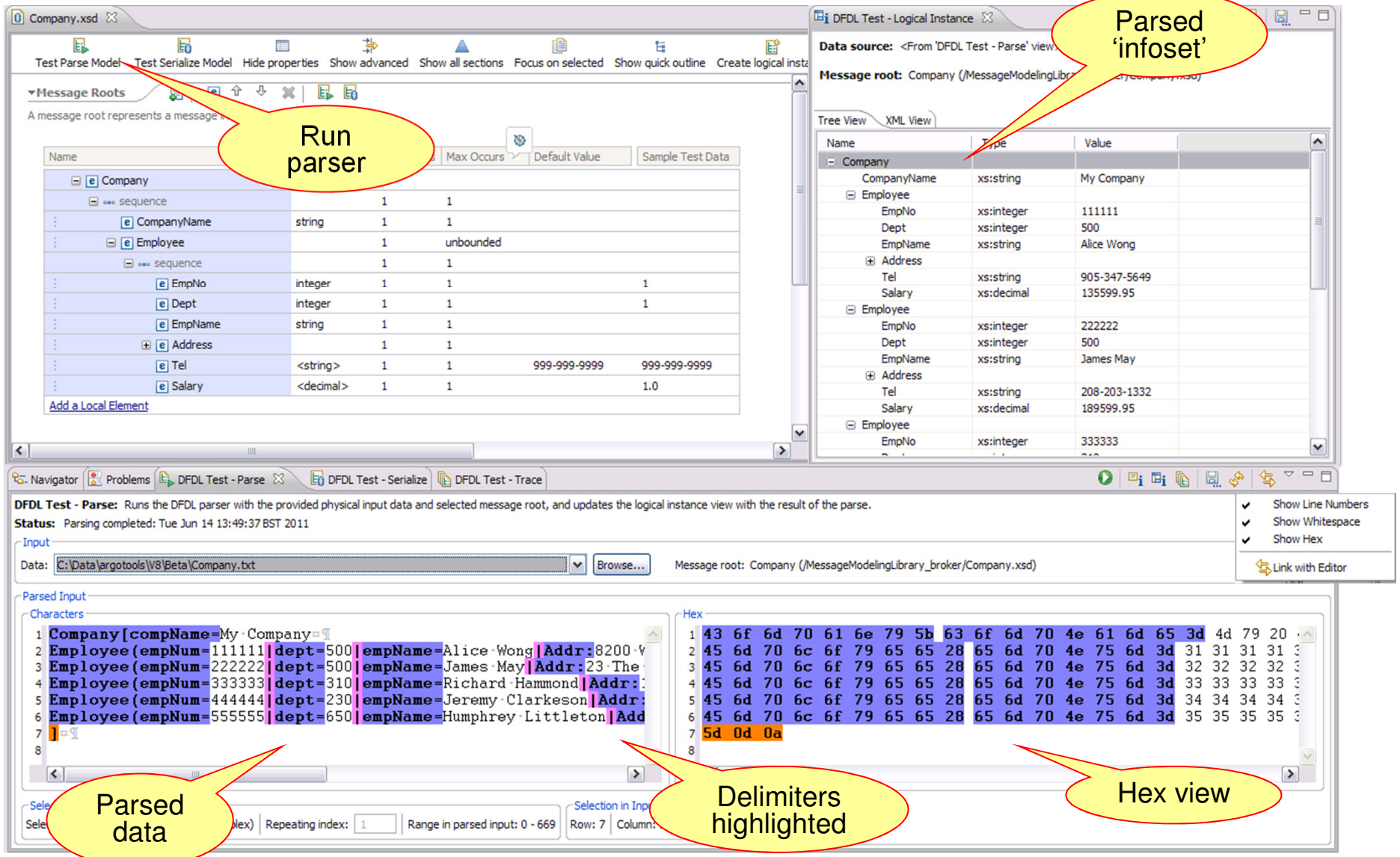
Agenda

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

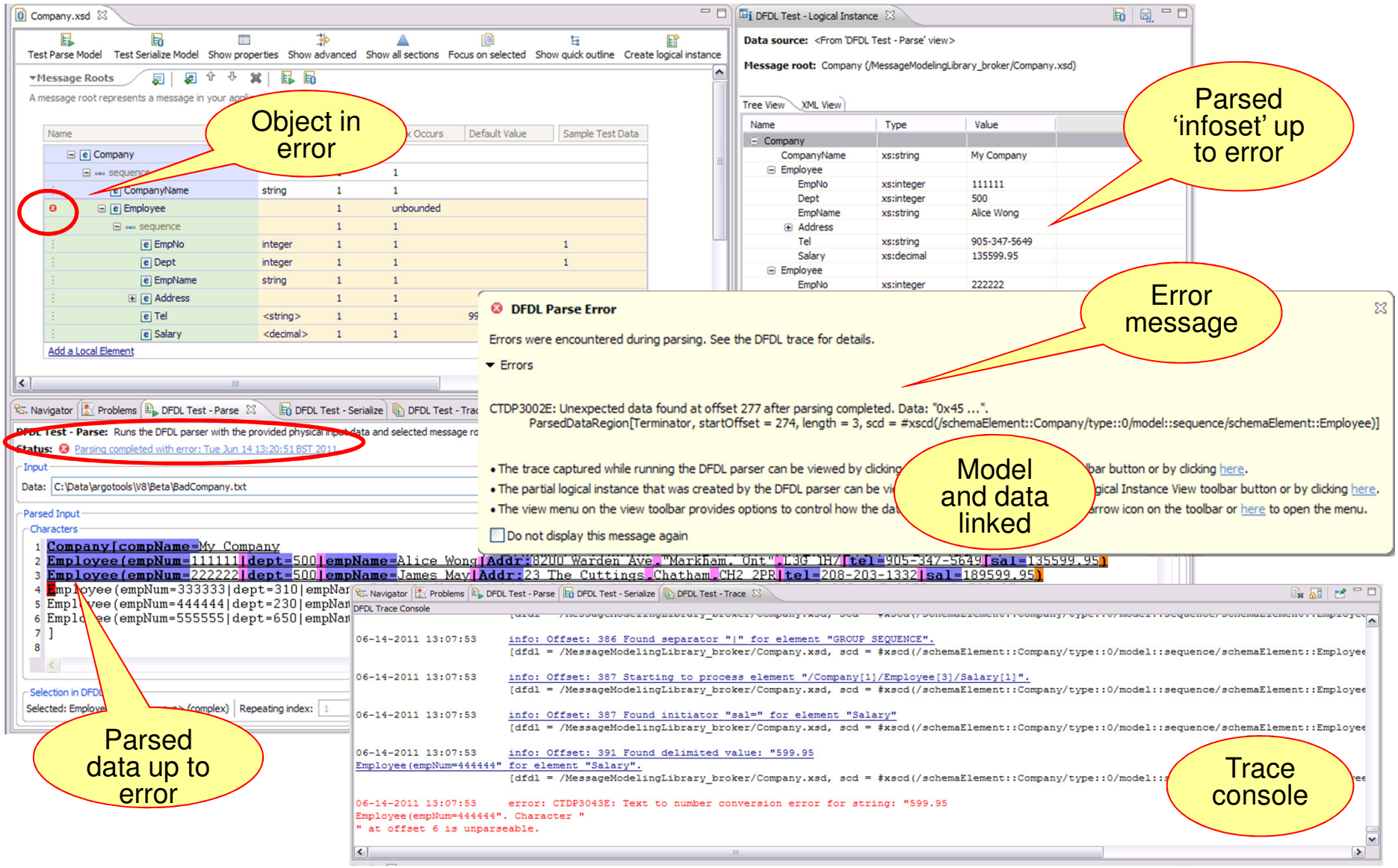
IBM DFDL 1.0 – Schema editor



IBM DFDL 1.0 – Schema debugger



IBM DFDL 1.0 – Schema debugger



Agenda

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

Next Steps

- DFDL-WG conference calls held every Tuesday
- Imminent spec redraft with errata folded in
- Public comment period
- Need to complete Tutorials
- Contributors welcome to open source ‘Daffodil’ project
- Join the mailing list http://www.ogf.org/mailman/listinfo/dfdl-wg

Links

- OGF DFDL home page: http://www.ogf.org/dfdl/
- DFDL 1.0 specification (pdf): http://www.ogf.org/documents/GFD.174.pdf
- DFDL 1.0 specification (html): http://www.ogf.org/dfdl/spec.php
- DFDL tutorial: http://redmine.ogf.org/dmsf/dfdl-wg?folder_id=5485
- DFDL-WG project: http://redmine.ogf.org/projects/dfdl-wg
- DFDL Wikipedia page: http://en.wikipedia.org/wiki/DFDL
- IBM WebSphere Message Broker developer edition: http://www.ibm.com/developerworks/downloads/ws/wmbd/
- Open Source Daffodil: http://opensource.ncsa.illinois.edu/confluence/display/DFDL
- DFDL Schemas on GitHub: https://github.com/DFDLSchemas

धन्यवाद
Hindi

多謝
Traditional Chinese

Grazie
Italian

ขอบคุณ
Thai

Спасибо
Russian

Obrigado
Brazilian Portuguese

Merci
French

Gracias
Spanish

شكراً
Arabic

多谢
Simplified Chinese

Danke
German

நன்றி
Tamil

ありがとうございました
Japanese

감사합니다

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