GWD-I
Category: INFORMATIONAL
GGF Data Format Description Language Working Group
Martin Westhead, EPCC, University of Edinburgh
et al…
10th September 2003

Data Format Description Language – Primer

Status of This Memo

This memo provides information to the Grid community regarding the specification of a Data Format Description Language. The document is currently an early draft which does not represent a consensus within the group. It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Abstract

The development of standards for describing format, structure and semantic content of data is essential for its automatic manipulation within a grid. The Data Format Description Language (DFDL) is a Global Grid Forum working group which draws on multiple existing format description languages and aims to develop an extensible standard for describing these features of data formats.

This document is a primer which is intended to give an overview to the motivations and illustrate how the various DFDL standards can be used in practice.

Contents

1. Introduction

To today's Internet infrastructure, the data that it moves around is just so many bits. The infrastructure is pretty good at ensuring that the order and integrity of the bits is maintained as they are moved. However, the information about the structure, format and meaning of those bits, which is essential to do anything with them, is primarily embedded implicitly in the applications that read the data.

As we build the Grid, the next generation of Internet infrastructure, it is becoming increasingly clear that the infrastructure will need to understand more about the structure and semantics of the bits that it is manipulating.

The grid community is developing a range of technologies that will allow the results of a computational job to be sent to a consumer with complete transparency over where or even when that job was executed. It seems clear that the consumer should be able to correctly interpret the results regardless of the byte-order those results are written in, or even what coordinates system they might use. In today's grid where the consumer is almost always a human being with additional knowledge about the data this information is rarely critical, because the human ensures necessary conversions and constrains are applied to the data. However, as we move to increased automatic data manipulation knowing about the data representation becomes essential.

Some Grid standards groups have explicitly identified this problem. The Data Access and Integration Services (DAIS) Working Group [1] explicitly identify the need for "information preserving" data movement capabilities and the Persistent Archives Research Group [2] require the ability to describe a digital entity in sufficient detail that it can be parsed and interpreted long after the application that created it has expired.

The grid community has also implemented multiple approaches to the quantification of this type of knowledge. The DODS (Distributed Oceanographic Data System) [3] evaluates the knowledge within a web interface API, making it possible to retrieve a pressure array in a particular coordinate system. The DataCutter technology [4] applies the structural knowledge directly at a storage repository through filters. The HDF technology [5] provides aspects of both structural and procedural knowledge within their file headers and access APIs. A way to unify the descriptions of relationships within digital entities is needed to promote interoperability between such existing systems.

The Data Format Description Language (DFDL –pronounced daffodil) [6] is a Global Grid Forum standards activity which is building on a number of implemented description languages to provide a general and extensible platform for describing data formats.

2. Related work

There are three directly related pieces of work that take the approach outlined in this paper:

- Binary XML, (BinX) [7]
- Binary Format Description (BFD) [8] and
- Earth Sciences Markup Language (ESML) [9]

Binary Format Description (BFD), which was developed as part of the US Department of Energy-funded Scientific Annotation Middleware (SAM) project at the Pacific Northwest National Laboratory (PNNL). BFD is based on the eXtensible Scientific Interchange Language (XSIL)

ESML is another XML description system which has been developed at Information Technology and Systems Center University of Alabama in Huntsville. ESML, BFD and BinX are all similar in concept to DFDL. DFDL aims to go further in terms of extensibility and expressiveness.

One of the earliest pieces of work in this area is a system developed by IBM called EXPRESS [10] (data Extraction, Processing and REStructuring System). It supported access to a wide variety of data and restructuring of it for new uses. The system was driven by two very high level nonprocedural languages: DEFINE for data description and CONVERT for data restructuring. Program generation and cooperating process techniques were used to achieve efficient operation.

Another important data representation standard is STEP [11], STandard for the Exchange of Product model data, the unofficial name for the evolving IS0 standard 10303-Product Data Representation and Exchange. This aims to facilitate data/information exchange between CAD/CAM/CAE systems.

The External Data Representation Standard (XDR) is an IETF standard defined in RFC1832 [12]. It is a standard for the description and encoding of data in binary files. Like BinX it defines a series of data primitives. Unlike BinX the blocksize and byte order of these primitives is predefined.

The Hierarchical Data Format (HDF) project [13] is run by NCSA. It involves the development and support of software and file formats for scientific data management. The HDF software includes I/O libraries and tools for analyzing, visualizing, and converting scientific data. HDF defines a binary data format in which the data is represented. HDF also provides software that allows the conversion of (most) HDF files to a standard XML representation.

The MPI standard [14] defined calls to construct and manipulate data in platform and language independent ways. The standard also includes a data packing format for data exchange. MPI does not have any mechanisms for run-time discovery of data types of unknown messages and any variation in message content invalidates communication.

Portable Binary Input/Ouput (PBIO) [15,16] is a library and API that supports data meta-representation. Users register the structure of the data that they wish to transmit/store or receive/read and PBIO transparently masks the differences. To alleviate the increased communication costs associated with interpreted format conversion, PBIO uses dynamically generated binary code to interpret the data.

The Internet Inter-ORB Protocol (IIOP) is part of the CORBA standard [17] and forms the common basis for broad-scope mediated bridging. The role of a bridge is to ensure that content and semantics are mapped from the form appropriate to one ORB to that of another, so that users of any given ORB only see their appropriate content and semantics. The General Inter-ORB Protocol (GIOP) element specifies a standard transfer syntax (low-level data representation) and a set of message formats for communications between ORBs which may be located on very different architectures.

Abstract Syntax Notation number One (ASN.1) [18] is a ISO standard formal notation used for describing data transmitted by telecommunications protocols, regardless of language implementation and physical representation of these data, whatever the application, whether complex or very simple.

The WAP Binary XML Content Format [19] is a W3C specification which defines a compact binary representation of XML designed to reduce the transmission size of XML documents, allowing more effective use of XML data on narrowband communication channels. This is different from the others in that it is a compression technology for XML encoded data. It is included in this list just to point out that it is different.

The DFDL differs from each of these existing technologies in one or more of the following ways:

- It is agnostic about the format and structure of the data – many of these existing standards describe their own formats, here we just describe whatever format is being used.
- The description is in XML.
- The approach is unique in separating the structure of the data from its semantics and this separation is used to make the approach highly extensible..

A standard language for describing data

This paper proposes that the solution to this problem is to provide a general purpose language for describing the structure of digital entities and attaching semantic labels to components of those structures. The language would provide a standard system for detailed characterization of data types in files and streams.

For example we could define a structure composed of eight successive bits and label it with the name byte. Then we could define another structure as being composed of four successive bytes and label that with the name int. Furthermore we can attach an attribute label to the same structure byteOrder="bigEndian". The labels can be used by software libraries and tools (like those in BinX), which understand this description language and enable appropriate interpretation for this sequence of 32 bits.

The language can be expressed in a standard syntax, such as XML to make it easy to build supporting tools and libraries.

There are significant benefits to such a language. Once the structure of a digital entity has been made explicit in this way we can search the data with respect to that structure, extract pieces of it or describe transformations and translations. Moreover all of this can be supported by generic tools. A file described in this way becomes effectively a database.

For example, with information about the structure and predefined semantic labels, we can provide a logical XML view of a data file. This view does not necessarily have to be materialized but can provide:

- a basis for XPath queries to identify structures within it.
- a basis to allow transformations of the data to be described in XSLT.
- human readable versions of the data – or portions of it through automatic translation to XML

APIs that allow applications developers to read and write data as if it were XML (using DOM and SAX) but underneath there would be a much more efficient binary representation.

The approach is also appealing for data archiving because the meaning of the bits in a file can be preserved even when the applications that read that file are defunct. It can also provide a uniform way for annotating binary files with respect to their internal structure.

Why not just use XML/HDF5?

If all the data on the grid were formatted in a standard way there would be no need to provide a way for explicitly describing it. An obvious choice for such a language would be XML, combined with XML Schema or DTDs. This is already a standard and provides the same sort of labeled structural description that our approach aims at. However, it is clear that whilst XML brings a number of benefits it is not appropriate for all types of data. For example, data that are arranged in highly repetitive structures, such as large arrays or tables, gain little from the, highly redundant, addition of repeated XML markup. Moreover, the current generation of XML tools is unlikely to be able to handle such files efficiently.

There are alternative, general data formats. HDF is a popular example that provides a self described, efficient binary format which includes both structural and procedural information within their file APIs. It has existing multi-language support and can handle complex data structures.

However, whilst there may be increased use of standard formats such as these, we believe that it is naïve to assume that standard formats will solve the problem. It will always be the case that any particular format will have drawbacks for some applications. The users will not always have control over the writing of the data that they consume, and even if they do they will frequently have large amounts of legacy data to manage.

Standard formats will always be useful and important but history suggests that it is very unlikely that a single standard, or even a manageably small set of data standards can provide ubiquitous data representation. The development of a standard language for describing data formats can help to unify and provide interoperability between different representations.

Can a single language be flexible enough?

Data formats can be very complex and involved. Can a single language capture all that complexity and still be manageable? The key to achieving this is to develop a powerful core language with generic functionality that can be extended as required to meet the idiosyncratic needs of particular data sets.

DFDL provides a strong basis for extensibility by explicitly separating the description of the structure of the data from its meaning.

The data's structure is described in a hierarchical way using the structural description language. Each structural component can be named and have attributes attached to it. Separate ontologies are then defined, which have associated code libraries. These ontologies consist of vocabularies of names and attributes to which meaning is attached by describing what library operations can be performed on them and what the results of those operations will be.

For example, a developer might need to add a new high precision floating point representation. They could define a new structure called highPrecisionFloat. The structure of the new type could be represented in DFDL structural language, new methods of accessing and manipulating this type would need to be added to the standard API. These methods would be described in the associated ontology and implemented in additional plug-in libraries. The ontology would probably need to specify the behaviour of the new type when existing methods were applied to it, so there would be a way to define and implement a standard conversion from a highPrecisionFloat to a native floating point representation.

The structural description language itself is very flexible. It contains operators to describe:

- Simple sequence – providing support for C-like structures.
- Hierarchical composition of types
- Repetition where the number of repeats can be fixed, defined explicitly in the data, or terminated by the occurrence of a particular sequence of bits (characters).
- Conditionals based on data values – allowing for unions to be defined
- Pointers or other references within the data

There will, no doubt, be limitations that are not feasible to overcome within this framework. It is unclear, at this point, what those limitations are. However, the existing implementations, BinX, BFD and ESML which provide a subset of this functionality, are providing solutions in a sub-domain of this problem, demonstrating the viability and suitability of this approach.

3. Data Format Description Language

The aim of the DFDL working group within the GGF is to define an XML-based language, the Data Format Description Language (DFDL), for describing the structure of binary and character encoded (ASCII/Unicode) files and data streams so that their format, structure, and metadata can be exposed. This effort specifically does not aim to create a generic data representation language. Rather, DFDL endeavors to describe existing formats in an actionable manner that makes the data in its current format accessible through generic mechanisms.

The DFDL description, like BinX, sits in a (logically) separate file from the data itself. The description is hierarchical and characterizes the structure and semantically labels the underlying bits. That is to say:

- **Structure** – how bits are to be interpreted as parts of low-level data types (such as, integers, floats, strings) and how low-level types are assembled into scientifically relevant forms such as arrays
- **Semantic labeling** – how meaning is assigned to these forms through association with variable names and metadata such as units of measure, how arrays and the overall structure of the binary file are parameterized based on array dimensions, flags specifying optional file components, etc.

Further, if the data file contains highly repetitive structures, such as large arrays or tables, such a description can be very concise.

DFDL is an abstraction and generalization of BinX. In BinX the primitive types are embedded into the language and new types can be constructed only from these primitives. In DFDL the primitives and the user defined types are defined using the same kind of ontology description. This makes DFDL much more flexible.

3.1 Formal Semantics

In order to focus the discussion the first document the working group addressed was the construction of a formal semantics for the DFDL structural description language [12].

Key to the semantics is the idea of a structured binary sequence. Conceptually this involves the addition of nested parentheses to the underlying sequence of bits this binary integer:

    00000000 00001000 00010010 00001100

Could be given a structure like:

    [[00000000] [00001000] [00010010] [00001100]]

We can then attach labels naming, for example, the outer bracket an integer and the inner brackets as bytes.

Formally the types in the language are defined as sets of such structured sequences. The structural description language is then a set of operators (e.g. repetition of different kinds, references to the data etc.) for defining these sets. It is similar in expressibility to a regular expression language.

A user of the language has the full power of the language at their disposal when describing a data format which allows them to add new semantic concepts as they are needed.
For example the following definitions are a description of a comma separated value table. The notation is as follows: structured sequence sets are enclosed in square brackets, subsequences are separated by ‘;’, ‘:=’ is an assignment, ‘-’ is set subtraction, ‘.*’ represents zero or more repetitions, ‘::’ is concatenation for structured sequence sets. (See [12] for precise definitions).

```
valueChar   :=  char - ( lineFeed | comma )
field       :=  [ (valueChar.* ); comma]
finalField  :=  [valueChar; lineFeed]
row         :=  ( (field.* )::[finalField] )
table       :=  ( row.* )
```

In this example, a “valueChar” is character that does not contain a comma or a line feed. A “field” is a sequence of a “valueChar”s followed by a comma. A “finalField” is like a field except the terminating character is a lineFeed return. A “row” is a sequence of zero or more “fields” followed by a “finalField” and a table is a sequence of zero or more “rows”.

The definitions of “comma”, “lineFeed” and “char” are assumed. These are actually defined in the strawman primitives ontology.

3.2 XML representation and Ontologies

The next stage in the process will be to define how the structural description language would be represented in XML. A strawman of this document exists at time of writing. The XML representation of the above CSV table would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<dfdl xmlns="http://www.dfdl.org/2003/dfdl"
xmlns:prim="http://www.dfdl.org/2003/primitives"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.dfdl.org/2003/prim primitives.xsd">
   <definitions>
      <!-- This null definition just says that we know what char is already -->
      <define>
         <newType name="valueChar"/>
         <toBe>
            <exclude>
               <either>
                  <prim:lineFeed/>
                  <prim:comma/>
               </either>
               <from>
                  <prim:char/>
               </from>
            </exclude>
         </toBe>
      </define>
      <define>
         <newType name="field"/>
         <toBe>
            <sequence>
               <repeat number="unbounded">
                  <type name="valueChar"/>
               </repeat>
               <prim:comma/>
            </sequence>
         </toBe>
      </define>
      <define>
         <newType name="finalField"/>
         <toBe>
            <sequence>
               <repeat number="unbounded">
                  <type name="valueChar"/>
               </repeat>
               <prim:lineFeed/>
            </sequence>
```

```
        </toBe>
      </define>
      <define>
        <newType name="row"/>
        <toBe>
          <concatenate>
            <repeat number="unbounded">
              <type name="field"/>
            </repeat>
            <sequence>
              <type name="finalField"/>
            </sequence>
          </concatenate>
        </toBe>
      </define>
    </definitions>
    <description>
      <repeat number="unbounded">
        <type name="row"/>
      </repeat>
    </description>
</dfdl>
```

The working group will produce a document defining exactly what will be required in the description of a DFDL ontology. At this point it is expected that it will include

- A schema of new type names
- A DFDL structural description of the new types
- A definition of the way the standard API methods will treat the new types
- An API and behavioral definition of any new methods for manipulating the types
- Potentially a description in an appropriate ontology language defining relationships between types.

The working group is charted to develop an ontology for basic types (integers, floating point numbers, characters etc.) and simple structures (arrays, tables, strings etc.). Obvious extensions to the work would include the definition of ontologies for SQL types, XML Schema types or even metadata like S.I. units.

4. Examples and Use Cases

4.1 Text Array

This example is fairly similar to the comma separated value example in the previous section. However it introduces the use of data references. Suppose we have data in the following format:

```
/**********/
2 2
273.2 271.2
215.2 231.2
/*********/
```

Such that line one contains two integers which represent the number of rows and columns of the subsequent table. The table itself is space separated, and the file can use C-style comments at any point.

A possible description of this file would be as follows:

```
comment := [char[ '/' ]; char[ '*' ]; ( char.* ); char[ '*' ]; char[ '/' ]]
header := [int<varName="numberOfRows">; int<varName="numberOfColumns">]
field := [float; ( space | lineFeed )]
```

```
fixedRow := ( field.#/header/int[@varName=numberOfColumns] )
fixedTable := ( ( fixedRow | comment ).#/header/int[@varName=numberOfRows] )

( ( comment.? )::header::( comment.? )::fixedTable )
```

which, in XML, is:

```
<?xml version="1.0" encoding="UTF-8"?>
<dfdl xmlns="http://www.dfdl.org/2003/dfdl"
xmlns:prim="http://www.dfdl.org/2003/primitives"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.dfdl.org/2003/prim primitives.xsd">
   <definitions>
      <define>
         <newType name="comment"/>
         <toBe>
            <sequence>
               <prim:char>/</prim:char>
               <prim:char>*</prim:char>
               <repeat number="unbounded">
                  <prim:char/>
               </repeat>
               <prim:char>*</prim:char>
               <prim:char>/</prim:char>
            </sequence>
         </toBe>
      </define>
      <define>
         <newType name="header"/>
         <toBe>
            <sequence>
               <prim:int varName="numberOfRows"/>
               <prim:int varName="numberOfColumns"/>
            </sequence>
         </toBe>
      </define>
   <define>
      <newType name="field"/>
      <toBe>
         <sequence>
            <prim:float/>
            <either>
               <prim:space/>
               <prim:lineFeed/>
            </either>
         </sequence>
      </toBe>
   </define>
   <define>
      <newType name="fixedRow"/>
      <toBe>
         <repeat number="#/header/int[@varName=numberOfColumns]">
            <type name="field"/>
         </repeat>
      </toBe>
   </define>
   <define>
      <newType name="fixedTable"/>
      <toBe>
         <repeat number="#/header/int[@varName=numberOfRows]">
            <either>
               <type name="fixedRow"/>
               <type name="comment"/>
```

```
          </either>
        </repeat>
      </toBe>
   </define>
      </definitions>
   <description>
      <concatenate>
         <repeat number="zeroOrOne">
                <type name="comment"></type>
        </repeat>
        <type name="header"/>
        <repeat number="zeroOrOne">
                <type name="comment"></type>
        </repeat>
        <type name="fixedTable"/>
      </concatenate>
   </description>
</dfdl>
```

Put some examples and use cases in here.

5. Conclusion

It is essential for the automation of data manipulation on the grid that standards are developed for describing the structure, format and semantic content of the data. This paper described two related pieces of work which aim to address that need. BinX is an existing implementation in real applications which demonstrates feasibility and effectiveness of the approach. DFDL is a GGF standards effort that uses experience gained from BinX, and the community, to address a broader set of requirements. Over time we aim to see a these projects converge.

6. Security Considerations

There are no security considerations that we are aware of at this time.

Author Information

Martin Westhead, M.Westhead@epcc.ed.ac.uk, EPCC, University of Edinburgh. James Clerk Maxwell Building, Mayfield Road, Edinburgh EH9 3JZ, UK.

Glossary

DFDL – Data Format Description Language

Need some more terms in here…

Intellectual Property Statement

References

[1] N. C. Hong, A. Krause, S. Malaika, G. McCance, S. Laws, J. Magowan, N. W. Paton and G. Riccardi , "Grid Database Service Specification", February 16, 2003 http://www.globalgridforum.org/Meetings/ggf7/drafts/DAIS_GGF7StatementSpec.pdf

[2] Reagan W. Moore, Andre Merzky "Persistent Archive Capabilities" http://www.zib.de/ggf/data/pa/Docs/GGF8-PA-2-Capabilities-v5.doc

[3] T. Sgouros, "DODS User Guide, Version 1.11", May 22, 2003, http://www.unidata.ucar.edu/packages/dods/user/guide-html/guide.html"

[4] M. Beynon, R. Ferreira, T. Kurc, A. Sussman, J. Saltz, "DataCutter: Middleware for Filtering Very large Scientific Datasets on Archival Storage Systems", IEEE Symposium on Mass Storage Systems, 2000"

[5] "Goddard DAAC's Hierarchical Data Format (HDF), http://daac.gsfc.nasa.gov/REFERENCE_DOCS/HDF/gdaac_hdf.html"

[6] DFDL WG webpage http://www.epcc.ed.ac.uk/dfdl

[7] BinX project webpage http://www.edikt.org/binx

[8] SAM homepage http://collaboratory.emsl.pnl.gov/docs/collab/sam/

[9] ESML homepage http://esml.itsc.uah.edu/index.jsp

[10] EXPRESS: A Data EXtraction, Processing, amd REStructuring System EXPRESS – Nan C. Shu, Barron C. , R. W. , Sakti P. Ghosh and Vincent Y. Lum. TODS vol 2, No 2, 1997, pp134-174.

[11] STEP http://filebox.vt.edu/users/vkern/step.html

[12] XDR (IETF RFC 1832) http://www.faqs.org/rfcs/rfc1832.html

[13] HDF http://hdf.ncsa.uiuc.edu/HDF5/

[14] MPI standard http://www-unix.mcs.anl.gov/mpi/

[15] Portable Binary Input/Output (PBIO) http://www.cc.gatech.edu/systems/projects/PBIO/

[16] Greg Eisenhauer, Karsten Schwan and Patrick Widener, "Efficient Wire Formats for High Performance Computing, Fabian Bustamente", proceedings of SC'2000.
http://www.cc.gatech.edu/systems/papers/schwan/Bustamante00EWF.ps

[17] CORBA Interoperability Overview http://www.omg.org/docs/formal/02-06-48.pdf

[18] Introduction to ASN.1 http://asn1.elibel.tm.fr/en/introduction/index.htm

[19] WAP Binary XML Content Format W3C NOTE 24 June 1999 http://www.w3.org/TR/wbxml/

[20] Astrogrid project webpage http://www.astrogrid.org/

[21] VOTable webpage http://www.us-vo.org/VOTable/

[22] FITS webpage
http://heasarc.gsfc.nasa.gov/docs/heasarc/fits.html

[23] OGSA-DAI webpage http://www.ogsadai.org/

[24] DFDL structural description language (GGF8 Draft)
http://www.epcc.ed.ac.uk/dfdl/DfdlStructuralDescription.pdf