Representing Scientific Data on the Grid with BinX – Binary XML Description Language

Martin Westhead, Mark Bull
EPCC, University of Edinburgh,
JCMB, Mayfield Road,
Edinburgh, EH9 3JZ, UK.
M.Westhead@epcc.ed.ac.uk, M.Bull@epcc.ed.ac.uk

Abstract

This paper outlines work-in-progress on a proposed new XML Schema standard: Binary XML description language (BinX), which provides the ability to describe the physical representation and the overall structure of arbitrary binary data files.

After considering the issues involved in the representation of Scientific Data sets in Grid environments we concluded that although XML can provide a very useful mechanism for representing metadata, it is often inappropriate for representing large scientific datasets themselves. However, there is a need for a standard way to describe binary datasets and to that end we have developed BinX. BinX and its associated libraries will support platform independent binary data, the construction of arbitrary data transformations and potentially, the ability for the application developer to work with their data as if it were in XML whilst retaining efficient manageable underlying binary representations.

The construction of basic libraries to support access to BinX data is well underway. We have also developed JAJA (Java Access to Just-about-any Array), a simple prototype browser for binary array files to demonstrate how the standard might be used. Finally, we mention existing standards that could support the construction of self-describing files that would include an XML metadata file description, as well as the dataset itself.

1. Introduction

XML is clearly today's standard of choice for the representation and exchange of structured data, particularly where that data must be read and interpreted by different applications written by different groups. XML and XML Schema provide a convenient, potentially human readable, easily extensible representation standard. It is tempting to assume, therefore, that all data exchanged on the Grid would be exchanged as XML. However, for many users[1] in the scientific community the prospect of producing output data as XML presents many disadvantages and offers very few opportunities.

The datasets for many scientific users are stored in very large (tens of gigabytes) regularly structured binary files, often one or more large arrays or tables. They have tools for reading and manipulating these files, often written in languages like Fortran with primitive file handling capabilities. Whilst it is possible, in principle, to provide XML representations of such data it is not clear why you would want to. An XML representation would have a number of drawbacks:

- The XML representation would be significantly (around 2-4 times) larger than the simple binary representation and therefore take longer to write, transport etc.
- Inappropriate representations: The proposed standard representation for a multidimensional array in XML to effectively build a tree of lists (everything in XML is a tree). This is a poor representation for scientific users because commonly required operations such as extracting a slice or a diagonal becomes difficult to do.

XML also present few advantages:

- Extensibility – The extensibility of XML is not enormously useful in this case. XML is most useful when the data is represents is richly structured. The simple arrays and tables that we are looking at do not have those features and are unlikely to change their in their basic representations.

[1] The BinX team have had requirements discussions with a number of eScience communities including Astrogrid, MyGrid, RealityGrid and QCDGrid.

- Readability – A 10Gb array is not very human readable. It can obviously be visualised with the right software but representing it as XML does nothing to improve this situation.
- Available tools – The available XML tools have not been designed for efficiently parsing such large files and their scalability may be severely tested.

It seems unlikely, therefore, that users with such datasets will represent them in XML. However, there is enormous value, and corresponding interest, in representing the metadata associated with the data in XML. The metadata will typically describe such things as how the data was produced (parameters, algorithms used etc), when and by whom. It would be very useful if that metadata could also contain a standard, canonical description of the structure and representation of the data itself. The work in progress, reported on here, is a straw man proposal for an XML Schema and supporting software libraries to address that need. The approach taken is described in the Section 2.

Related work

One of the earliest pieces of work this area was a system developed by IBM called EXPRESS [1] (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 [2], 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 standardization effort begun in 1984 and was joined by PDES (Product Data Exchange using STEP), an American standardization initiative being developed by the IGES/PDES Organization. The first set of International Standard documents was approved in 1994. Information modeling, supported by the language EXPRESS, addresses information about a product's entire life cycle.

The External Data Representation Standard (XDR) is an IETF standard defined in RFC1832 [3]. It is a standard for the description and encoding of data in binary files. It differs from BinX in that:

- It defines aspects of the data encoding. BinX is intended to describe any (most) encodings rather than specify features of the encoding that should be used.
- BinX is XML based.

The Hierarchical Data Format (HDF) project [4] 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 also, however, 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.

2. *BinX*

BinX is the Binary XML description language. Its aim is to provide a canonical description for data stored in binary files. Once we have carried out some initial groundwork we propose to take the work forward as a GGF standard.

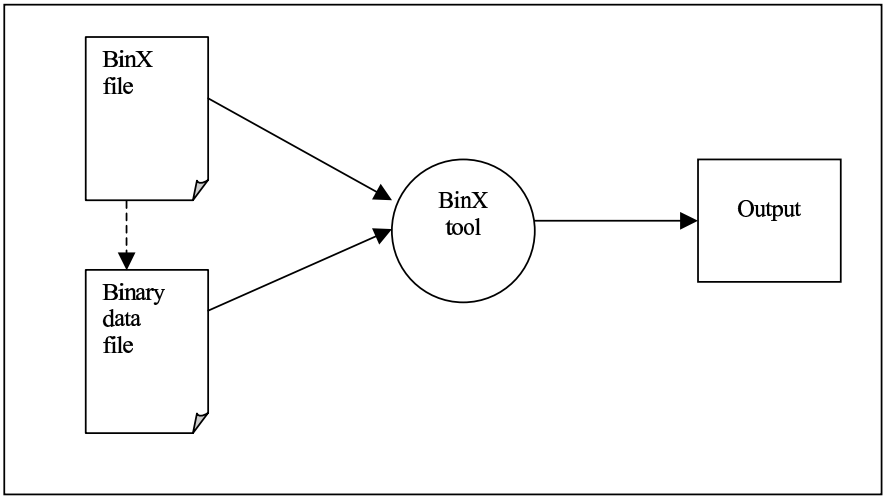


Figure 1. Showing how the BinX file is used in to describe the format of the binary file.

Figure 1 is intended to illustrate how the BinX file could be used in practice. The BinX file describes the structure and format of a binary data file and can also contains a URL which points to that file. This allows the construction of tools that can read a very wide range of file formats. Such tools could be presented as Web services and could have functionality to convert between formats, to extract pieces of the data (e.g. slices or diagonals of an array), or to browse the file. JAJA, discussed below, is a BinX tool that provides simple browsing functionality of an array described in BinX.

BinX provides the ability to describe three levels of features in a binary file:

1. The underlying physical representation (e.g. bit/byte ordering)
2. The primitive types used (e.g. IEEE float, integer)
3. The structure of the data itself (e.g. array, list of fields, table)

In many applications we anticipate that BinX will be included as part of application specific metadata.

The representation of data in binary files is much more standard than it used to be. The prevalence of the IEEE floating point standard [5] has simplified a number of the issues. From the point of the physical representation it is still necessary to specify:

- The byte ordering – big-endian/little endian
- The bit ordering – big-endian/little endian (although this is almost always big endian)
- Blocksize – many binary formats pad out data fields so that they are always a multiple of a given block size.

BinX provides for the representation of a broad range of different primitive types. Including all those that can be represented in XML Schema [6]. We anticipate that the standards process will eventually bring this to an appropriate representative set.

In terms of structural representations we have been guided by the work on XDR. Anything that can be represented in XDR can be represented in BinX so we have provision for variable and fixed length arrays, structs, strings, unions etc. We have also made provision for the description of data streams.

The definition of BinX includes a "typeDef" mechanism that allows the user to define/rename new types. The intention in our design is to try to provide the minimum number of basic types and then to provide an include file that defines a set of standard extensions.

```
<?xml version="1.0" encoding="UTF-8"?>
<dataset            xmlns="http://http://schemas.nesc.ac.uk/binx/binx"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://http://schemas.nesc.ac.uk/binx/binx
binx.xsd"
byteOrder="bigEndian" bitOrder="bigEndian" blockSize="32">
  <definitions>
    <typeDef typeName="complexType">
      <struct>
        <ieeeFloat-32 varName="real"/>
        <ieeeFloat-32 varName="imaginary"/>
      </struct>
    </typeDef>
  </definitions>
  <file src="http://www.epcc.ed.ac.uk/testFile.bin">
    <ieeeFloat-32 varName="inputParameter1"/>
    <integer-32 varName="inputParameter2"/>
    <arrayFixed>
      <defType typeName="complexType"/>
      <dim indexFrom="0" indexTo="99" name="x"/>
      <dim indexFrom="0" indexTo="4" name="y"/>
    </arrayFixed>
  </file>
</dataset>
```

Figure 2. *An example BinX file.*

Figure 2 shows a small simple BinX file. The root tag is <dataset> which is can contain a set of type definitions, contained within the <definitions> tag, followed by one or more file descriptions contained within the <file> tag. In this example we can see that a new type "ComplexType" has been declared in the definitions section that is defined to be a struct containing two floats, one called "real" and one called "imaginary". There is only one file in this example, which is located at:

`http://www.epcc.ed.ac.uk/testFile.bin`

This file contains two numbers a float (inputParameter1) followed by an integer (inputParameter2) followed by a two dimensional array of our new complex number type.

Notice that the <dataset> tag allows us to define the byte order, the bit order and the blocksize. These can be changed for individual files, or indeed for individual fields.

Areas of application

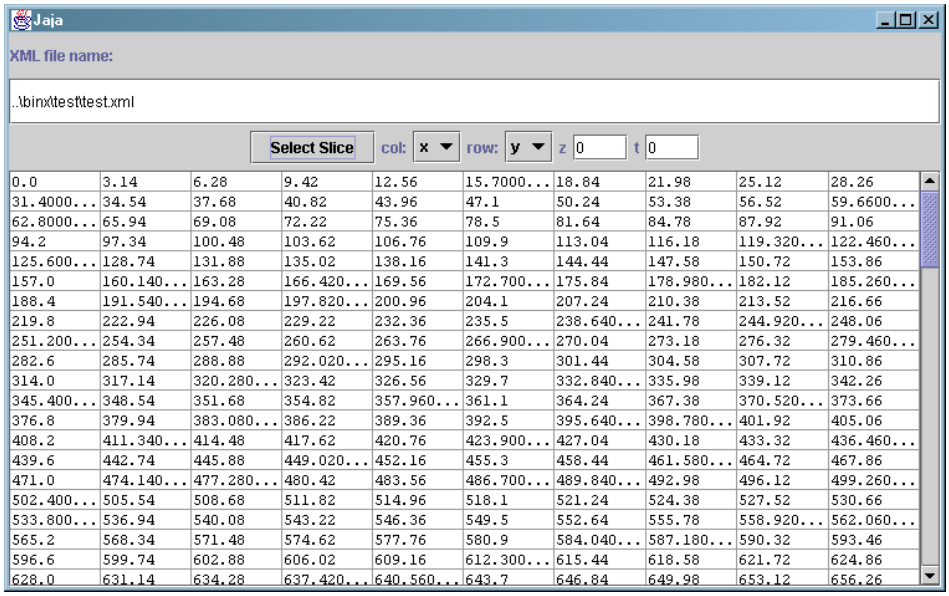
We foresee the following ways in which BinX can be used:

1. Platform independence – once data is described by BinX it should be possible to read and write it (using the BinX library) on any platform regardless of the native configuration of the file system.
2. Data transformations – the library will provide a very convenient basis for building arbitrary transformations between binary data formats.
3. XML without the tags – once the data has been described in BinX it will become possible to expose it to the application level as if it were in XML (i.e. as SAX events). Thus users can have the benefit of working with the standard XML tools and APIs without needing to pay the cost in the size of their data.

Current status

BinX is currently being developed as part of the eDICT project (http://www.edikt.org/) at the National eScience Centre in Edinburgh. At the time of writing we have a relatively stable version of the schema and a C++ library capable of reading the simple data types the next stages will be the reading of more complex data types and then the ability to write out both BinX files and the data itself. This will provide the functionality required for the first area of application, platform independent files, and provide the basis for building examples of the second, transformations.

3. JAJA



0.0	3.14	6.28	9.42	12.56	15.7000...	18.84	21.98	25.12	28.26
31.4000...	34.54	37.68	40.82	43.96	47.1	50.24	53.38	56.52	59.6600...
62.8000...	65.94	69.08	72.22	75.36	78.5	81.64	84.78	87.92	91.06
94.2	97.34	100.48	103.62	106.76	109.9	113.04	116.18	119.320...	122.460...
125.600...	128.74	131.88	135.02	138.16	141.3	144.44	147.58	150.72	153.86
157.0	160.140...	163.28	166.420...	169.56	172.700...	175.84	178.980...	182.12	185.260...
188.4	191.540...	194.68	197.820...	200.96	204.1	207.24	210.38	213.52	216.66
219.8	222.94	226.08	229.22	232.36	235.5	238.640...	241.78	244.920...	248.06
251.200...	254.34	257.48	260.62	263.76	266.900...	270.04	273.18	276.32	279.460...
282.6	285.74	288.88	292.020...	295.16	298.3	301.44	304.58	307.72	310.86
314.0	317.14	320.280...	323.42	326.56	329.7	332.840...	335.98	339.12	342.26
345.400...	348.54	351.68	354.82	357.960...	361.1	364.24	367.38	370.520...	373.66
376.8	379.94	383.080...	386.22	389.36	392.5	395.640...	398.780...	401.92	405.06
408.2	411.340...	414.48	417.62	420.76	423.900...	427.04	430.18	433.32	436.460...
439.6	442.74	445.88	449.020...	452.16	455.3	458.44	461.580...	464.72	467.86
471.0	474.140...	477.280...	480.42	483.56	486.700...	489.840...	492.98	496.12	499.260...
502.400...	505.54	508.68	511.82	514.96	518.1	521.24	524.38	527.52	530.66
533.800...	536.94	540.08	543.22	546.36	549.5	552.64	555.78	558.920...	562.060...
565.2	568.34	571.48	574.62	577.76	580.9	584.040...	587.180...	590.32	593.46
596.6	599.74	602.88	606.02	609.16	612.300...	615.44	618.58	621.72	624.86
628.0	631.14	634.28	637.420...	640.560...	643.7	646.84	649.98	653.12	656.26

Figure 3

JAJA (Java Access to Just-about-any Array) is a prototype BinX tool, built to demonstrate the potential of the work. The JAJA interface (Figure 3) can be used to display arbitrary slices through a multidimensional array specified in BinX.

4. Self describing files

An important requirement that came originally from the Astrogrid team was that it should to be possible to include the description of a binary file in the file itself. If the description of the file is located in a different file there is a danger that the correspondence between the two files could be lost (as they are copied/moved around) and then the data would be rendered effectively useless.

There are a number of approaches that could be adopted to solve this problem. A number of ad-hoc methods are frequently employed for this purpose, such as making the first four bytes of the file be an integer representing the offset in bytes to the start of the data etc. However this is a general problem and there are existing standards that can be applied. The most relevant of which is DIME which is designed to provide a way of attaching binary data to XML files, such as SOAP messages. DIME [7] is a simple, lightweight message format that encapsulates multiple messages with header information (such as MIME type) that allow the individual messages to be later extracted using a DIME parser.

5. Future directions

The next steps for this work will be to bring it to review within the standards processes of the Global Grid Forum. In the immediate future we aim to work on:

- Tools and libraries for reading and writing data files described using BinX.
- Testing activities with our user communities to investigate whether BinX can capture all that is required.
- A Java GUI for writing BinX descriptions so that users are not forced to write the XML by hand.

6. Acknowledgements

The work reported here was started at EPCC in the University of Edinburgh and is currently ongoing within the eDICT project at the UK National e-Science Centre.

7. References

[1] 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.

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

[3] http://www.faqs.org/rfcs/rfc1832.html

[4] HDF

[5] "IEEE Standard for Binary Floating-Point Arithmetic", ANSI/IEEE Standard 754-1985, Institute of Electrical and Electronics Engineers, August 1985.

[6] http://www.w3.org/TR/xmlschema-2/

[7] H. Nielsen, H. Sanders, E. Christensen, " Direct Internet Message Encapsulation (DIME)", INTERNET DRAFT "http://www.ietf.org/internet-drafts/draft-nielsen-dime-01.txt", Microsoft, May 2001. This is work in progress.