INTERNATIONAL TABLES FOR CRYSTALLOGRAPHY

ISSN 1574-8707
it.iucr.org

Volume I, X-ray Absorption Spectroscopy and Related Techniques
ISBN: 978-1-119-43394-1

Developing specifications for XAFS information interchange

James R. Hester*

Australian Nuclear Science and Technology Organisation, Locked Bag 2001, Kirrawee DC, NSW 2232, Australia.
*Correspondence e-mail: james.hester@ansto.gov.au

A number of data formats in widespread use outside the XAFS community are suitable for comprehensive descriptions of XAFS experiments. The suitability of each data format for particular use cases is assessed.

Keywords: information interchange.

1. Introduction

As information cannot depend on the way in which it is arranged in a data file, it is always possible to cleanly divide a data-interchange specification into two parts: the *ontology* and the *format*. The ontology defines the scientific meanings of items found in data files. Each individual meaning is labelled for convenience with a *data name*. The format describes the location and arrangement of the values associated with these data names in a data file. A single ontology can therefore be used with multiple file formats, and the ontology developed for use with a particular format can be repurposed and extended for other formats.

The ontologies underlying traditional XAS data formats are simple. A format consisting of a header followed by the absorption spectrum presented in a single table is typically sufficient to capture values for all data names. Notable examples of these formats include XAFS Data Interchange (XDI) and the ICAT XAFS database input format (Asakura *et al.*, 2018). However, as made clear by Sarangi (2018) and Trevorah *et al.* (2019), much richer metadata describing both the experiment and data-modelling process are required in order to properly process, model and assess modern XAS experiments. Recording data values described by ontologies that have been expanded to cover richer metadata requires the adoption of a more complex data format, as discussed in the next section.

A survey of possible data-transfer formats for XAS was presented by Ravel *et al.* (2012). Of the surveyed formats, only the XDI format has seen much subsequent use. A textual XAS Interchange Format (XIF) for biological XAS later outlined by Sarangi (2018) allows more than one table per data file and as such is more suited to a larger ontology.

The following sections briefly assess the way in which a broad-coverage XAFS ontology might be matched to a representative sample of scientific file formats. Each file format is assessed both in terms of its suitability for a scientific ontology and according to other desirable practical features of an interchange format. The analysis below may readily be extended to the hundreds of information-interchange formats that are not covered here.

2. Ontological considerations

The suitability of a particular file format may be evaluated from an ontological perspective according to the arguments of Hester (2016) based on the olog formalism of Spivak & Kent (2012). In this formulation, a data name is a function mapping values from some domain to some codomain, with some data names simply containing domain values. The values taken by a data name in a data file are a record of particular associations between domain and codomain values, and a data name will in general take multiple values in a single data file. A file format is able to encapsulate data from any ontology if (i) an arbitrary number of data names can take multiple values, (ii) the type of data values appearing in the ontology can be represented by that format and (iii) the correspondence between data-name values and the domain values from which they were mapped is preserved. The data formats considered below all satisfy these ontological criteria. Traditional XAS formats generally allow a single table only, so that it becomes impractical to include an arbitrary number of multi-valued data names and therefore these formats fail meet the ontological criteria.

In common with most scientific data, the data types encountered in EXAFS can be covered by integers, real numbers, text and multi-dimensional array data. Data of these types may always be preserved textually using ASCII-encoded bytes, technically satisfying (ii), although native provision for these types in the chosen format is desirable in order to reduce specification and programming effort when mating the ontology to a given format. For example, while multi-dimensional arrays may always be stored as a sequence of numbers together with array dimensions, index order and optional compression type, a format which defines arrays as a fundamental type allows data programmers to ignore array representation details and to instead rely on format-specific programming libraries.

(iii) is generally trivial to satisfy in any format offering ordered sequences of values. If data names taking multiple values are stored as arrays, the position in the array can be used to link corresponding domain and codomain values. Alternatively, a common pattern in hierarchical formats is to distribute values for a single data name across multiple nodes: in this case, values for a data name appearing at a lower level in the hierarchy correspond to the particular node value that they lie under.

3. Practical considerations

Desirable practical features of an interchange format typically include portability, long-term accessibility, efficient storage, rapid access and convenience (Hinsen, 2012). The particular use case will determine the relative weight to assign each of these features.

Archival use requires that the data files are likely to be accessible over the long term. This means that at some nominated future time both the storage media and file system are readable, and that the exact standard according to which the data were stored in the files is available, preferably with accompanying software tools. A simple heuristic for estimating the likely longevity of a standard is to assess the number of data files currently written according to that standard and any standards upon which it depends: the more data files that are in circulation, the longer it will take for that standard to become irrelevant and thus unsupported. For relatively new standards, the widespread availability of unencumbered specifications, open-source, cross-platform software tools and independence from any single organization are reasonable leading indicators of longevity.

The most long-lived and widespread standard for information storage in files is the ASCII encoding of text, suggesting that standards based on text files using this encoding are the safest way of encapsulating archival data. Text encoded according to the more recent ASCII-compatible UTF-8 Unicode standard (Allen, 2015) is becoming increasingly ubiquitous; this is therefore also a reasonable choice. Binary formats are worth consideration where measures have been taken to ensure that file-reading software, or at least file-format specifications, will remain accessible over the long term. For example, the complete text of the relevant specifications and source code for file-reading software may be preserved in the same directory as any archived data files.

Storage efficiency becomes a consideration when large amounts of data need to be handled. In general, binary formats lead to smaller files: floating-point approximations to real numbers usually require fewer bytes on average compared with their text equivalent, and bytes in a binary file can cover the whole range of bit patterns found in compressed data, whereas text files are restricted to the 95 printable ASCII characters[1]. A requirement for rapid access also favours binary files, as numerical values can be stored in a computer-ready form, rather than requiring the relatively time-intensive process of converting a sequence of characters into floating-point numbers or integers. Binary files also usually include directory and indexing structures, facilitating the rapid retrieval of data values and portions of large arrays.

Convenience of file access is important when direct human interaction with the file contents is common. Reasons for such access include the need to add additional metadata or simply to examine data items of interest. If custom processing of data files is likely to be widespread, the availability of software tools also becomes important. Text files have generally been the most convenient form for both human and software access, due to the universal presence on all modern operating systems of familiar editors, tools and libraries that work with ASCII, and increasingly UTF-8, encoding. The comparative inconvenience of binary formats may be offset by the provision of libraries, editors and tools which understand and leverage the file structure in ways that generic text tools do not.

Table 1 assesses each of the above practical aspects for a selection of file formats. Further details of each format are provided in the next section.

[1] While this inefficiency can be rectified by compressing text files, this then leads to issues in accessing software to uncompress files over the long term and largely negates the advantages of using plain text.

Table 1
Comparison of a selection of file formats for scientific use.

A format is considered convenient if it is either a text format or a variety of tools and libraries exist to easily manipulate the file contents. Full multi-dimensional array support requires that users of standard access tools do not need to routinely consider or implement array compression, dimensions and storage order.

Format	Text/binary	Rapid access	Convenience	Multi-dimensional arrays	Efficient storage
CIF	Text	No	Yes	Partial	No
HDF5	Binary	Yes	Partial	Yes	Yes
XML	Text	No	Yes	No	No
JSON	Text	No	Yes	Partial	No
SQLite	Binary	Yes	No	No	Yes

4. File formats

4.1. CIF

The Crystallographic Information Framework (CIF; Hall & McMahon, 2006) encompasses two text file formats, known as CIF1 and CIF2 (Bernstein *et al.*, 2016). A CIF file is an ASCII (CIF1) or UTF-8 (CIF2) encoded text file composed of a set of data blocks. Each data block contains an arbitrary number of arbitrarily ordered key–value pairs and loops. Loops are notional columns of data listed together to form tables. Real numbers can be recorded to arbitrary precision, and images may be encoded efficiently as a text string according to the imgCIF specification (Bernstein, 2006). CIF files are ideal for archiving, but the format is relatively poor in terms of efficient access when files contain of the order of tens of megabytes or more (for example, a file with multiple images), because the whole file must potentially be read sequentially in order to find a single data item.

Ontological data names map directly onto CIF data names, and values for the data names appear in the file either as a single value or as columns in a loop. Simple examples of XAFS CIF files are provided in Ravel *et al.* (2012) and the supplementary information of Trevorah *et al.* (2019). Note that these files contain unofficial CIF data names for XAFS data, and that no CIF data names for use with XAFS have been defined at the time of writing.

4.2. HDF5

An HDF5 file (The HDF Group; http://www.hdfgroup.org/HDF5/) is a binary structure within which data items are arranged in a logical hierarchy. Leaf elements in the hierarchy can be arbitrarily dimensioned arrays, and attributes can be attached to both the leaf elements and to the nodes in the hierarchy. Software libraries for accessing HDF5 files have been released under an open-source licence and the file-structure specification is also freely available, suggesting that the readability of archived files over the long term is guaranteed, particularly if copies of the source code and specification are preserved together with the archived files. HDF5 files have excellent storage properties for image data, due to built-in image-compression algorithms and rapid access to stored data based on internal directory structures.

Various choices exist for mapping data names into the HDF5 structure. A viable minimalist approach would place all data names in the top level of the data file, eschewing use of the hierarchical structure while benefiting from the storage and access efficiency of the file format. Alternatively, some storage savings can be obtained in a situation where multiple data names take many repeated values, with a single group of repeated values for one data name matching one or more groups of repeated values for another data name. In such a case, creating hierarchies in which each level of the hierarchy corresponds to a single data name, and each node value to a single value for that data name, will lead to reduced repetition. This approach becomes more economical with increasing numbers of 'key' data names; that is, data names upon which the value of other data names depend. An example is provided in Hester (2016).

The HDF5-based NeXus standard (Könnecke *et al.*, 2015) includes two specifications for XAS data: NXxas (raw data) and NXxasproc (for processed data). Neither has seen significant adoption. A generic HDF5 'Data Exchange' specification has also been published (De Carlo *et al.*, 2014), but requires additional specification of XAFS-specific data names.

4.3. XML

The XML format is a text-based format derived from the HTML format used to describe the layout of web pages (W3C, 2008). Like HDF5, XML files are arranged in hierarchies, with attributes that can be attached to the nodes and leaves of the structure. A rich set of tools is available to manipulate, validate and transform XML files. XML has no native type for the compact representation of arrays of numbers; an array is instead constructed by creating a sequence of single leaf values, each appearing in a node of the same type and level as the other members of the array, with the order of appearance in the file serving as the position in the list. Although XML can thus be manipulated to satisfy our key ontological criterion (i), the absence of a native list data structure creates extra programming work and requires considerable storage space, with at least seven extra characters per list element required. As a well supported ASCII file format, XML is suitable for archiving but is undesirable in terms of storage and access efficiency.

4.4. JSON

JSON (short for Javascript Object Notation) is a text-based format typically used to transfer data between web applications (Internet Engineering Task Force, 2014; ECMA International, 2013). JSON allows the construction of hierarchies and has native lists and arrays. Due to its widespread use in web applications, many tools are available. Considerations of storage and transfer efficiency are similar to those for CIF.

4.5. SQLite

SQLite is a widely deployed programming library for creating and accessing a relational database (The SQLite

Consortium, 2016). Unlike many database systems, an SQLite database does not require a separate server process to access data, and the database data are stored in a single file that can be archived and transmitted. The database is a binary file, with support for floating-point, integer, text and binary 'blob' datatypes. When the blob datatype is used to store images, separate data names describing the image parameters must be defined and processed.

References

Allen, J. D. (2015). Editor. *The Unicode Standard Version 8.0 – Core Specification*, pp. 124–130. http://www.unicode.org/versions/Unicode8.0.0/ch03.pdf.

Asakura, K., Abe, H. & Kimura, M. (2018). *J. Synchrotron Rad.* **25**, 967–971.

Bernstein, H. J. (2006). *International Tables for Crystallography*, Vol. G, edited by S. Hall & B. McMahon, pp. 199–205. Dordrecht: Springer.

Bernstein, H. J., Bollinger, J. C., Brown, I. D., Gražulis, S., Hester, J. R., McMahon, B., Spadaccini, N., Westbrook, J. D. & Westrip, S. P. (2016). *J. Appl. Cryst.* **49**, 277–284.

De Carlo, F., Gürsoy, D., Marone, F., Rivers, M., Parkinson, D. Y., Khan, F., Schwarz, N., Vine, D. J., Vogt, S., Gleber, S.-C., Narayanan, S., Newville, M., Lanzirotti, T., Sun, Y., Hong, Y. P. & Jacobsen, C. (2014). *J. Synchrotron Rad.* **21**, 1224–1230.

ECMA International (2013). *The JSON Data Interchange Format*, 1st ed. https://www.ecma-international.org/wp-content/uploads/ECMA-404_1st_edition_october_2013.pdf.

Hall, S. & McMahon, B. (2006). *International Tables for Crystallography*, Vol. G. Dordrecht: Springer.

Hester, J. R. (2016). *Data Sci. J.* **15**, 12.

Hinsen, K. (2012). *Comput. Sci. Eng.* **14**, 70–74.

Internet Engineering Task Force (2014). *The JavaScript Object Notation (JSON) Data Interchange Format.* https://tools.ietf.org/html/rfc7159.

Könnecke, M., Akeroyd, F. A., Bernstein, H. J., Brewster, A. S., Campbell, S. I., Clausen, B., Cottrell, S., Hoffmann, J. U., Jemian, P. R., Männicke, D., Osborn, R., Peterson, P. F., Richter, T., Suzuki, J., Watts, B., Wintersberger, E. & Wuttke, J. (2015). *J. Appl. Cryst.* **48**, 301–305.

Ravel, B., Hester, J. R., Solé, V. A. & Newville, M. (2012). *J. Synchrotron Rad.* **19**, 869–874.

Sarangi, R. (2018). *J. Synchrotron Rad.* **25**, 944–952.

Spivak, D. I. & Kent, R. E. (2012). *PLoS One*, **7**, e24274.

The SQLite Consortium (2016). *SQLite Documentation*. https://www.sqlite.org/docs.html.

Trevorah, R. M., Chantler, C. T. & Schalken, M. J. (2019). *IUCrJ*, **6**, 586–602.

W3C (2008). *Extensible Markup Language (XML) 1.0.* https://www.w3.org/TR/REC-xml/.