The NeXus Data Format

Mark Könnecke,[1] Frederick Akeroyd,[2] Herbert J Bernstein,[3] Aaron S. Brewster,[4] Bjoern Clausen,[5] Stephen Cottrell,[6] Jens Uwe Hoffmann,[7] Pete Jemian,[8] David Männicke,[9] Raymond Osborn,[10] Peter F. Peterson,[11] Tobias Richter,[12] Jiro Suzuki,[13] Bejmanin Watts,[14] Eugen Wintersberger,[15] and Joachim Wuttke[16]
(NeXus International Advisory Committee)
[1)] *Laboratory for Development and Methods*
Paul Scherrer Institute
5232 Villigen-PSI
Switzerland
[2)] *ISIS, Rutherford Appleton Labortaory, UK*
[3)] *imgCIF, Dowling College USA*
[4)] *Lawrence Berkeley National Laboratory, USA*
[5)] *Los Alamos National Laboratory, USA*
[6)] *ISIS, Rutherford Appleton Laboratory, UK*
[7)] *Helmholtz Zentrum Berlin, Germany*
[8)] *Advanced Photon Source, USA*
[9)] *ANSTO Australia*
[10)] *Argonne National Laboratory, USA*
[11)] *Spallation Neutron Source, USA*
[12)] *Diamond Light Source, UK*
[13)] *KEK, Japan*
[14)] *Swiss Light Source, Paul Scherrer Institute, 5232 Villigen-PSI, Switzerland*
[15)] *DESY, Germany*
[16)] *FRMII, JCNS, Germany*

(Dated: 8 August 2014)

NeXus is an effort by an international group of scientists to define a common data exchange format for neutron, muon and x-ray experiments. NeXus is built on top of the scientific data format HDF-5 and adds domain specific rules for organizing data within HDF-5 files as well as a dictionary of well defined domain-specific field names. The NeXus data format has two purposes. First, NeXus defines a format that can serve as a container for all relevant data associated with a beamline, an increasingly important function. Second, NeXus defines standards in the form of *application definitions* for the exchange of data between applications. NeXus provides structures for raw experimental data as well as for processed data.

Keywords: NeXus, data format, HDF-5, x-ray, neutron,data analysis, data management

I. INTRODUCTION

Increasingly, major neutron and x-ray facilities choose to store their data in the NeXus data format. Since 2006, NeXus[1] has undergone substantial refocusing, refinement and enhancement as described in this paper.

Historically, neutron and x-ray facilities choose to store their data in a plethora of home-grown data formats. This scheme has a number of drawbacks addressed by NeXus:

- It makes the life of the traveling scientist more difficult then it needs to be as they have to deal with multiple files in different formats, file converters and such in order to extract scientific information from the data.
- An unnecessary burden is imposed on data analysis software producers as they have to accommodate so many different formats.
- The whole idea of open access to data is sabotaged if the data is in a format which cannot be easily understood.
- Modern high speed detectors produce data at such a rate that many older single image storage schemes become impractical and an efficient container format is a necessity.

The first necessity for a data format is a physical file format: how is the data written to disk? Rather than inventing yet another format, NeXus chose HDF-5[2] as the binary container format. HDF-5 is efficient, self describing, platform independent, in the public domain and well supported by both commercial and free software tools.

NeXus adds to HDF-5:

- Rules for organizing domain-specific data within a HDF-5 file
- A dictionary of documented domain-specific field names
- Definitions of standards that can be validated

The development of NeXus is overseen by a committee, the NeXus International Advisory Committee (NIAC).

II. NEXUS DESIGN PRINCIPLES

NeXus encourages its users to store all relevant data associated with their experiment or data analysis in a

NeXus container file. This includes the state of the beamline, detector data, metadata, sample environment logs and more. The general idea is for users to have to read only one logical file. That enables the user to analyse data in ways yet unforeseen because the necessary additional fields have been stored. The burden of writing complete files is usually with the producing facility or software, not the user. Thus this is no problem for the user.

Consistent with the container file objective, it is always possible to add data, even non-NeXus data, to a NeXus data file without breaking NeXus.

Of course, NeXus strives for platform independence, a self describing efficient file format, public domain specifications and tools. These aims were the reasons for choosing HDF-5 as the physical file format.

A NeXus container file may contain hundreds of fields. Often, all that is needed for data reduction or analysis are a small fraction of these - perhaps fewer than 20 fields. These necessary fields are dependent on the experimental technique and NeXus provides a way to describe a set of necessary fields as a NeXus application definition. Files can be validated against such an application definition by checking for the presence of the required fields. This is the way in which NeXus expresses file standards.

III. NEXUS FILE HIERARCHIES

NeXus data files are organized into a hierarchy of groups which, in turn, can contain further groups or fields, very much like an internal file system. For an overview of the NeXus data file structure for raw experimental data see Table I.

TABLE I. Overview of the structure of a NeXus raw data file

NXroot			
The root level of a NeXus file	NXentry		
	All data belonging to one scan or run. A given NeXus file can contain multiple related scans or runs	NXinstrument	
		The data needed to describe an instrument. Contains groups for each relevant instrument component	NXsource
			NXcollimator
			NXattenuator
			NXdetector
			. . .
		NXsample All the information about the sample	
		NXmonitor Intensity monitor	
		NXuser User information	
		NXdata Links to plottable data in the NXdetector group – one instance for each detector bank. This provides support for generating a typical plot automatically	

In the following sections, we will describe some of the rules that define the overall structure of NeXus files. Many fine details of the NeXus format have been thoroughly discussed and are now well defined, but for the sake of brevity, we will not present an exhaustive view of NeXus here. A full listing of NeXus rules are given in the NeXus manual[3]. Some examples of these additional rules include those governing:

- How axes are associated with data
- That units must be given with the data
- How data is to be stored in NeXus fields
- How to describe array data which is not in ANSI C storage order

A. NeXus Raw Data File Hierarchy

The major focus of NeXus has been the recording of "raw" experimental data, i.e. information taken directly from the experimental equipment, or processed only as required to provide physically meaningful values. The NeXus raw data file hierarchy is the consequence of some practical considerations. When looking at a beamline it is easy to discern different components: beam optic components, sample position, detectors and such. It is quite natural to mirror this physical separation with a logical arrangement of storing the data from each component in a separate group. This approach explains the list of beamline components in the NXinstrument group presented in TableI. As there can be multiple instances of the same kind of equipment, like slits or detectors, in a given beamline it becomes necessary to add type information to the group name, which is provided by the NeXus class name. By convention NeXus class names start with the prefix NX. Each NeXus group describing a beamline component contains further groups and fields describing the component. A field can contain a single number, a text string or an array, as appropriate to the data to be described.

The requirement to store multiple related scans or runs in the same file or to capture a complete workflow in a file causes the beamline component hierarchy to be pushed one level deeper into an NXentry group in the hierarchy. The NXentry group thus represents one scan or run (or a processed data entry, as will be discussed later). The NXentry group also holds the experiment metadata, such as the date and time at which it was performed.

The requirement to access sample information and counting information such as monitors quickly pulled this information out of the beamline hierarchy and into the NXentry level.

Another requirement was to provide easy access to a default visualization of the data. This idea was realised

TABLE II. Overview of the structure of a NeXus raw data file for an instrument with multiple methods

NXroot			
...	NXentry		
	All data from multiple methods	NXinstrument	
		...	NXsource
			...
		NXsample	
		NXuser	
		NXdata	
		NXsubentry SAS specification	
		NXsubentry powder diffraction specification	

with the NXdata type group at NXentry level. The NXdata group contains the data and axes data necessary to create a default plot of the data in the entry. This is data which will typically live in the NXdetector or other groups in the beamline component list. HDF-5 links are used to avoid duplicating data. A link is a pointer to data sitting at another place in the file hierarchy, like a symbolic link in a unix file system. The NXdata group is intended to hold such links to the relevant pieces of information.

1. Multiple Method Instruments

Particularly at X-ray sources, some beamlines implement multiple techniques in the same instrument. For example small-angle scattering (SAS) and powder diffraction in the same beamline are sometimes measured simultaneously (such as SAXS/WAXS). This poses a problem for an automatic data analysis tool to find its data for processing. NeXus solves this problem through a scheme involving these steps:

1. All the data is stored in one NXentry hierarchy

2. At entry level, for each experimental method of the beamline an NXsubentry group is introduced. The NXsubentry groups have the same hierarchy as the NXentry group. NXsubentry holds only the data relevant for the experimental method it describes. In order to avoid data duplication, links into the main NXentry hierarchy are used. NXsubentry usually only holds the 20-30 data items required by its corresponding NeXus application definition.

These steps are represented in the overview of Table II.

2. Scans

Scans come in all shapes and sizes. Almost anything can be scanned against anything. An additional difficulty is that in practice, the number of scan points in the scan cannot be known in advance since it is possible that a scan may be interrupted or terminated before its planned number of observations. Thus, it is a challenge to standardize a scan. NeXus solves these difficulties through a couple of conventions and the use of a HDF-5 feature called unlimited dimensions. With the HDF-5 unlimited dimensions feature, one axis of the data is allowed to expand without limit and the size of a data array does not need to be declared in advance. Data can be appended to an array along the unlimited dimension as required.

Scans are stored in NeXus following these conventions:

- Each variable varied or collected in the scan is stored at its appropriate place in the NeXus beamline hierarchy as an array. The array's first dimension is the number of scan points. This is the unlimited dimension in the implementation and data is appended at each scan point to the array.

- The NXdata group holds links to all the variables varied or collected during the scan. This creates something equivalent or better than the tabular representation people are used to for scans. The main detector data scan be plotted against any scanned parameter as well as against everything that was deemed worth recording in addition to that, reading the NXdata group alone.

NeXus allows multi dimensional scans too. This makes it very simple to produce meaningful slices through data volmes even with NeXus-agnostic software (like hdfview). Interrupting a multi-dimensional scan may, depending on the software used, leave some of the data in an uninitialised state (usually the HDF-5 fill value). This is behaviour is currently undefined in NeXus.

3. Coordinate System and Positioning of Components

For analysing data it is often necessary to know the exact position and orientation of beamline components. The first thing needed is a reference coordinate system. NeXus chose to use same coordinate system as the neutron beamline simulation software McStas[4].

For describing the placement and orientation of components, NeXus stores the same information as used for the same purpose in the Crystallographic Interchange Format (CIF)[5]. CIF (and NeXus) stores the details of the translations and rotations necessary to move a given component from the zero point of the coordinate system to its actual position. As coordinate transformations are not commutative, the order of transformations must also be stored.

B. Processed Data

At the request of the user community, NeXus created a simplified structure for storing the result of data processing: be it reduction or analysis. In many cases even the reduced data is big enough to need an efficient binary representation. A good example is a tomography reconstruction. A tabular representation of the NeXus processed data file structure is given in Table III.

TABLE III. Overview of the structure of a NeXus processed data file

NXroot			
	NXentry		
		NXprocess	
		The data needed to describe this processing step	input:NXparameter
			output:NXparameter
The root level of a NeXus file	All data belonging to this processed data entry	NXsample	
		All the information about the sample	
		NXdata	
		The result data from the data processing including its axes	

The hierarchy is much reduced as it is not important to carry all experimental information in the data reduction. In contrast to the raw data file structure, NXdata in the processed file structure is the place to store the results of the processing, together with its associated axes if the result is a multi-dimensional array.

Information about the sample and instrument can be stored in NXsample and NXinstrument groups as required.

In addition, there is a new structure to store details about the processing such as the program used, its version, the date of processing, and other metadata in the NXprocess group. The NXprocess group can hold additional NXparameter groups which are containers for storing the input and output parameters of the program used to perform the processing.

IV. THE NEXUS DICTIONARY

As can be seen from the discussion of the NeXus file hierarchy, NeXus arranges data into groups which have a type descriptor, the NeXus base class, associated with them. For each of these NeXus base classes, there exists a description of all the fields and groups possible within such a NeXus base class as well as definitions of the scientific meaning of each field. The collection of these NeXus base classes constitute the NeXus dictionary. The term class is a little misleading: NeXus base classes are not classes in strict object oriented notation, but are dictionaries of allowed names. A common misconception is that NeXus users have to give values for all the names in a NeXus base class. This is not the case; only those applicable to the instrument and situation at hand need to be written to the file.

The NeXus base classes are encoded in NeXus Description Language (NXDL)[3]. NXDL is just another application of XML. Thus a NXDL file is an XML file that specifies the content of the NeXus base classes.

Procedures are in place to extend NeXus base classes quickly when necessary.

V. NEXUS APPLICATION DEFINITIONS

The NeXus approach to standardization is unique for a standard. At first NeXus asks you to store as much relevant information about your data as possible. In a second step NeXus then defines a standard for a certain use case of NeXus, an application definition, wherein domain-specific fields for that use case are described as well as their locations within the NeXus file. This definition is the NeXus application definition. Use cases can be experimental techniques like SAS or powder diffraction, while processed data use cases such as tomography reconstructions or S(Q,Omega), or even administrative use cases such as archiving. This scheme works because data in a NeXus file can be searched for. Thus it is trivial to ignore data which is not of interest for a particular use case.

Another way to look at a NeXus application definition is as a contract between file writers and file consumers about the minimum content of the file necessary to fulfill that particular use case as well as a map of where the data should be located and its format.

NeXus application definitions are expressed in NXDL. They may be parsed either by humans or by software and they may be validated for syntax and content. The NXDL files are used to validate the structure of NeXus data files. A tool exists to perform such validation.

A great number of candidate NeXus application definitions exist which were derived from our understanding of the technique described. For each of these, the NeXus team seeks community approval. Currently scientists from NeXus and the IUCr are nearly finished with a NeXus application definition for macromolecular crystallography. CBFlib[6] is being extended to work with NeXus-MX format. This work will be published in another paper. Work on another NeXus application definition for reduced small-angle scattering data is also in progress[7] by members of canSAS, NeXus, and the IUCr Commission on Small-Angle Scattering.

VI. UPTAKE OF NEXUS

NeXus is already in use as the main data format at facilities like Soleil, Diamond, SINQ, SNS, Lujan/LANL and KEK. Other facilities like ISIS, DESY and the muSR community are in the process of moving towards NeXus as data format. At LBNL, NeXus is currently being adapted for XFEL serial crystallographic data.

The adoption of NeXus took time. The reason is that NeXus is often chosen whenever a facility starts operation or undergoes major refurbishments. For those facilities where there is an existing and working pipeline from data acquisition to data analysis, the resources are usually lacking to move towards NeXus.

This is reflected in the experience of the muon community. For the ISIS source, the move to a Windows PC-based data acquisition system in 2002 required a new data format, providing an ideal opportunity to exploit the emerging NeXus standard[8]. In contrast, sources at PSI, TRIUMF and KEK continued to make good use of existing formats and software. More recently, funding from the EU has enabled the community to develop the Application Definition as a common exchange format for muon data[9]. Whether used as the main or an intermediate format, users will soon be able to produce compatible NeXus files for data written across all facilities, enhancing the uptake of NeXus within the community.

VII. NEXUS GOVERNANCE

The development of NeXus is overseen by the NeXus International Advisory Committee (NIAC). In the NIAC, most relevant facilities are represented. It is easy to join the NIAC if there is justified interest.

VIII. BACKWARDS COMPATIBILITY

In the past, NeXus supported data files in HDF-4, HDF-5, and XML file formats. To support writing software to write and read these file formats, the NeXus application/programmer interface (NeXus-API) was provided. This API still exists and is maintained at a bug fix level. On request of the community, we now concentrate our efforts only on the HDF-5 files and tools.

IX. SUMMARY

NeXus has matured considerably over the last 10 years and is now in use in many facilities. NeXus is flexible enough to accommodate a wide variety of instruments and scientific applications. Yet it is efficient enough to handle the data coming from modern high speed detectors. For more information, do not hesitate to consult the NeXus WWW–site[10] or to contact the members of the NIAC.

[1] S. Campbell, J. Cadogan, M. Furusaka, N. Hauser, M. James, R. Osborn, and C. Wilson, eds., *Proceedings of the Eigth International Conference on Neutron Scattering: ICNS 2005*, Physica B Condensed Matter, Vol. 385-386 (Elsevier, 2006) the State of the NeXus data format.

[2] Hdfgroup, *HDF-5* (2014 (accessed july 2014)), `http://www.hdfgroup.org/HDF5/`.

[3] NIAC, *NeXus Manual* (2014 (accessed july 2014)), `http://download.nexusformat.org/kits/definitions/nexus-manual-3.1.0.tar.gz`.

[4] P. Willendrup, E. Farhi, and K. Lefmann, Physica B **350**, 735 (2004).

[5] S. Hall and B. McMahon, *International Tables for Cystallography Volume G: Definition and exchange of crystallographic data* (Wiley, 2006).

[6] H. J. Bernstein and P. J. Ellis, in *International Tables For Crystallography*, Vol. G: Definition and Exchange of Crystallographic Data, edited by S. R. Hall and B. McMahon (Springer, Dordrecht, NL, 2005) Chap. 5.6, pp. 544 – 556, see `http://sf.net.projects/cbflib` and `http://www.bernstein-plus-sons.com/software/CBF/`.

[7] canSAS, *Description of the canSAS2012 data format* (2014 (accessed july 2014)), `http://www.cansas.org/formats/canSAS2012/1.0/doc/`.

[8] P. K. D. Flannery, S.P. Cottrell, Physica B **326**, 238 (2003).

[9] S. P. Cottrell, F. Pratt, A. Hillier, P. King, F. Akeroyd, A. Markvardsen, N. Draper, Y. Yao, and S. Blundell, Physics Procedia **30**, 20 (2012).

[10] NIAC, *NeXus A common data format for neutron, x-ray and muon science* (2014 (accessed july 2014)), `http://www.nexusformat.org/`.