Mantid Workspace Storage in NeXus Format

Ronald Fowler, STFC Rutherford Appleton Laboratory

Introduction

The Mantid project is developing a framework for data analysis for neutron and muon data within the STFC ISIS department. This requires the ability to both read existing data files in NeXus format and to be able to write the workspace data out in a similar file format. NeXus is a general format for describing X-ray, neutron and Muon data which uses either HDF or XML to store the information. Several DTDs have been, or are in the process of being, developed to cater with specific types of instruments. For example there are DTDs for Time of Flight (TOF) RAW data[4], Muon data[1] and processed data [2]. These DTDs build on the available fields defined by NeXus to specify the data which is required or optional for a given area.

Mantid makes use of two dimensional "workspaces". These are populated with data loaded from an ISIS Raw data file or from a NeXus data file but may be processed using a variety of algorithms. Initial processing will require details of the detector geometry, but the final workspace results may not need this. Hence Mantid needs a format which may reduce to processed data [2] including details of applied algorithms, but which also has the option of containing detector geometry information and environmental data. This document describes the format used to store Mantid workspace data and extensions to the NXshape to allow more accurate representations of objects.

Saving workspace data to a NeXus file

A Mantid workspace may be saved to either a new NeXus file or as an additional entry within an existing file. If the workspace is written as an additional entry in the source NeXus file then it is possible to make links to the existing NXsample and NXinstrument sections for the sample and instrument data. This is not possible if the data was read from an ISIS Raw file, or if a new NeXus file is being written.

For a new NeXus file an NXroot section will be created. The section will follow the format given in [1], with the "creator" field set as "Mantid". Entries will be labeled as "Mantid_workspace_<n>" where <n> is a sequence number starting from 1. The internal Mantid workspace name will be stored within the entry as the title field.

The format of the NXentry for a Mantid workspace will be based upon the NXprocessed DTD for processed data with some additions. In all cases of NX_FLOAT Mantid use NX_FLOAT64.

NXprocessed and IXmatidworkspace

The NXprocessed DTD can be used to store some types of Mantid Workspace data. It is intended to represent data that has been processed so that it no longer needs the instrument definition to interpret it. NXprocessed is not sufficient to describe the typical Mantid workspaces since they may retain and use information about the instrument. The NXprocess section is a collection of NXnote sections which will be used to store information about the processing steps used to create the current data.

An extended form of NXprocessed is defined which allows storage of all the data necessary to describe a Mantid workspace. Since this is not identical to NXprocessed and is not an official format, it will be referred to as IXmantidworkspace.

The IXmantidworkspace format will have the following fields:

NXentry

Re	Name	Type	Value	Description	Attributes
1	title	NX_CHAR	e.g. gemws	Mantid workspace name	
1	definition	NX_CHAR		mantidworkspace	URL=*path* Version=1.0
0/1	instrument	NXinstrument		Description of instrument.	
0/1	instrument_source	NX_CHAR	e.g. Gem.xml	Path to file describing instrument	date=iso8601 Version=*n*
0/1	Instrument_xml	NX_CHAR		Mantid XML description of instrument	
1	sample	NXsample		A copy of the sample related data within the Mantid workspace.	
1	data	NXdata		Workspace data, including errors.	
1	process	NXprocess		Mantid algorithm steps used	

NXdata

Re	Name	Type	Value	Description	Attributes
0/1	Xvariable	NX_FLOAT[:]		Dimension scale defining X axis of the data in regular case.	long_name="{Axis label}" distribution="0|1" units=”{units}”
0/1	Xvariable	NX_FLOAT[:,:]		X axis values for ragged axis case	long_name="{Axis label}" distribution="0|1" units=”{units}”
1	Yvariable	NX_FLOAT[:]		Y axis values	long_name="{Axis label}" distribution="0|1" units=”{units}”
1	data	NX_FLOAT[:,:]		Data values. Mantid will use 2D spaces.	signal="1" axes="[...]" units=”{units}”
1	errors	NX_FLOAT[:,:]		Standard deviations of data values.	

The main difference with NXprocessed is the addition of data on the instrument. For the case of an instrument that is described by a Mantid XML document, the file name of the XML document is stored in an NXnote section. The detector geometry may also be written into an NXdetector section or the actual Mantid XML description can be saved as instrument_xml. Storing the XML ensures that the

For the case of ragged bin boundaries between spectra, the X-axis variable will be written as a 2D array, rather than a 1D one.

NXsample

The NXsample section contains details of the sample and its environment. Within Mantid this will include the name of the sample material along with properties and time series properties read from the input file. These properties can include the sample temperature, magnetic field and sample size. Time series properties include log files, such as the temperature, magnetic field, etc. The set of log files will vary with the source of the data. On output from Mantid all log files will be written into the NXsample section.

Some properties generated by the algorithm will be ones that are not represented by a workspace, for example the output of a fitting algorithm might be a set of 3-tuples giving parameter names, values and errors. These are not dealt with in the current output format.

NXinstrument

An NXinstrument entry may be used to store data on the instrument. This will include the sections of NXdetector for the spectra-detector mapping and also the approximate geometry of the detectors themselves. Mantid uses a non-Nexus XML file to describe the instrument geometry. This contains accurate details of the detectors and their configuration within the instrument. Constructive solid geometry (CSG) [5] is used, similar to the format described in [3]. At present NeXus does not allow the

range of objects that can be described in CSG in its set of NXshape values, so the data written to the NeXus detector description may not be accurate. An extension of the NXshape definition is described which would allow CSG object definition in NeXus. To store the accurate geometry, either the filename of the Mantid XML can be stored, or the complete XML definition can be written as a text block within the Nexus file.

NXprocess data

A field similar to NXprocess is used to store the actual data. The data fields will differ from the data read from the input file, with the addition of an error values and subsequent processing. Below we discuss how a Nexus NXprocess entry can be used to store data from algorithm history section along with the data fields.

Workspace storage in NXdata format

The Mantid workspace is stored in an NXdata section within an NXprocess entry, as defined in the above table. The default format assumes that the data is two dimensional on a grid with regular axes in the two dimensions.

Within a workspace there will be two sets of values to write, the data and the error. These are written as two separate sets of values within the NXdata section. The data has the attribute “signal=1” and the name “data” while the error data is labelled “errors”. For the case of each spectrum having the same X-axis coordinates two 1D arrays are used to define X and Y values. Otherwise the X-coordinates need be written as a 2D array of the same shape as the data and errors.

The Mantid axes names need to be stored in this section along with the values on the axes. Typically these may be TOF along the spectra and, if unprocessed, spectra number along the other axis. The set of names allowed will be as defined within Mantid. The current set is:

- TOF (microseconds)
- Wavelength (Angstrom)
- Energy (meV)
- dSpacing (Angstrom)
- MomentumTransfer (1/Angstrom)
- QSquared (Angstrom^-2)
- DeltaE (meV)
- DeltaE_inWaveNumber (1/cm)

These will follow the NeXus convention of using just the *units* attribute to describe the actual unit (angstrom, meV, etc.). The attribute *long_name* will be used to record the full name of the axis data.

Algorithm history storage in NXprocess format

To save algorithm history data in the Nexus NXprocessed format, the existing fields will be used. The NXprocess section of NXprocessed is defined as containing entries of type NXnote. These notes can take sequence numbers to define the order of the steps. The meaning of these fields is left to the application. For a Mantid workspace the NXprocess section will be:

NXprocess MantidSteps

Re	Name	Attribute	Type	Value	Description
1	environment		NXnote		The environment for Mantid
1+	MantidAlgorithm_{n}		NXnote		One note for each algorithm.

The following will be used to describe the application of a single Mantid algorithm, for example LoadRaw:

NXnote MantidAlgorithm_1

Re	Name	Attribute	Type	Value	Description
1	author		NX_CHAR		Mantid/<username>
1	date		ISO 8601		Date algorithm applied
0/1	type		NX_CHAR	Plain/text	
1	description		NX_CHAR	e.g. Mantid/LoadRaw	Algorithm name
1	data		NX_BINARY	e.g. Filename = GEM38770.raw OuputWorkSpace=w1	Parameter values for algorithm

The binary data is currently just a list of keyword value pairs. A more sophisticated XML description might be used in a future version. The same format is used for subsequent algorithmic steps, to produces the output workspace, with a separate NXnote for each. The optional parameters will also be written, to make the defaults explicit. The name of the note, *MantidAlgorithm_1,* contains the sequence number to identify the order in which the algorithms have been applied to generate the final workspace. The computational environment and software version is also written out in a separate NXnote. The environment values will be written as text in keyword/value pairs in a similar format to the algorithm data. The environment currently consists of the keywords *Version, OSname, OSversion* and *Username*.

NXnote MantidEnvironment

Re	Name	Attibute	Type	Value	Description
1	author		NX_CHAR		Mantid/<username>
1	date		ISO 8601		Date written
0/1	type		NX_CHAR	Plain/text	
0/1	description		NX_CHAR	Mantid environment data	
1	data		NX_BINARY	Version=1,…	Keyword pairs from Mantid environment

The algorithm data stored in this simple format is only meaningful within the context of the Mantid framework. While other users could follow the flow of the analysis by looking at the names and parameters used within algorithms, they may not be able to reproduce them exactly without having access to the details of the implementation used by Mantid.

Storing log data and parameter data

The exact content of the NXsample section varies between NeXus file definitions. For example there are significant differences between the two versions of NeXus file in [1]. Similarly there are differences between the Muon and TOF-RAW NeXus files.

The set of parameters and logs that are relevant within Mantid, and hence should be loaded from the original NeXus file into the workspace, is yet to be fully defined. In the present implementation all parameter value pairs (e.g. temperature=271.6K) and all log files loaded from the input NeXus file or from the ISIS RAW file will be saved within the NXsample section of the output Mantid file.

The basic set of parameters that Mantid is expected to load, if defined in the input file, includes:

- Sample name and description
- Proton charge (microAmp*hour)
- Sample temperature and log of temperature
- Sample magnetic field and log of field

For ISIS Raw file input a simple mapping will be made to convert ISIS log file names to meaningful values. Some ISIS Raw log files contain text values, rather than numeric values. These cannot be written as normal NXlog data as this does not support non-numeric values. The NXlog will be extended to allow NX_CHAR values.

Storing Instrument and Detector Information

The NXinstrument section will contain the fields:

NXinstrument instrument

Re	Name	Attibute	Type	Value	Description
1	name		NX_char		Name of instrument
1+	detector		NXdetector		Details of detectors

The spectra-detector mapping information needs to be recorded in the file. Two methods of recording this have been proposed, in the TOFRaw format [4] using NXdetector.unganged, and in [1] using a set of mapping tables. The TOFRaw method is an approved NeXus format and is directed at the problem of mapping from a hardware point of view. It assumes that detector information is given for the “ganged” detectors and stores the unganged data separately. This allows for plotting programs to directly display the ganged data since it includes the averaged parameters of the ganged detectors, as well as the unganged values. As Mantid does not store ganged data it is easier to use the format described in [1] within an NXdetector block to described detector to spectrum mapping.

NXdetector MantidDetector

Re	Name	Attibute	Type	Value	Description
1	spectrum_index		NX_INT32[ns]		List of global spectra
1	detector_index		NX_INT32[ns]		detector_index[j] points into detector_list giving start of detectors for jth spectra
1	detector_count		NX_INT32[ns]		Count of detectors for each spectra

1	detector_list		NX_INT32[nd]		List of detectors forming each spectrum.

Mantid may make use of the TOFRaw ganging scheme in future if this proves useful for efficiently saving large ISIS workspaces.

The detector geometry may also be written in this section. The NeXus TOFRaw format will be followed. This allows a general description as a series of point detectors, while also allowing a more efficient description in terms of linear or area detectors. The initial save format will just use the more general point layout format.

Re	Name	Attibute	Type	Value	Description
0/1	layout		NX_CHAR	point	How detector is represented
0/1	detector_number		NX_INT[:]		
0/1	polar_angle		NX_FLOAT[:]		
0/1	azimuthal_angle		NX_FLOAT[:]		
0/1	distance		NX_FLOAT[:]		
0/1	geometry		NXgeometry		

This will be extended in future to allow for linear and area detectors.

Extending the set of NeXus NXshape types

Mantid allows the definition of geometrical shapes using either a set of surfaces and CSG operations on them, or in terms of a small set of common finite objects that are easier to set up. Some of these shapes are similar to existing NeXus objects, but allow more general alignment without use of the translation and orientation sections. The finite shapes are:

- Mantid sphere (similar to NXsphere, but with an offset from the origin) defined by origin point **p** and radius *r*.

- Mantid finite cylinder (similar to NXcylinder, but with general orientation and offset) defined by height, radius, axis normal vector and location of base point. (**n,p**,*h,r*)
- Mantid cuboid (similar to NXbox, but with general orientation and offset) defined by set of four corner points. (**p1**,**p2**,**p3**,**p4**)
- Mantid hexahedron defined by eight corner points. (**p1**,**p2**,**p3**,**p4**,**p5**,**p6**,**p7**,**p8**)
- Mantid finite cone defined by normal vector along cone axis, vertex point, height and angle. (**n**,**p**,*h,a*)
- Mantid torus defined by normal vector, centre point, radius of torus centre and radius of torus tube. (**n**,**p**,*r,R*)

A set of infinite objects can also be defined, which are only useful when used within a CGS context. These are defined by infinite surfaces, but adding a sign to them generates a volume, as the points on one side of the surface. By default the sign is positive. The set of infinite shapes is:

- Mantid infinite plane defined by normal and point on plane. (**n,p**)
- Mantid infinite cylinder defined by normal, apex point and radius. (**n,p**,r)
- Mantid infinite cone defined by normal, apex point and radius. (**n,p**,a)
- Mantid general quadratic surface, defined by the 10 coefficients of a general quadratic surface equation in 3D. (a,b,c,d,e,f,g,h,j,k in `ax^2+by^2+cz^2+dxy+exz+fyz+gx+hy+jz+k=0`)

Some of the new finite objects could be represented using NXgeometry, but to retain the flexibility when using CSG, the new objects will be defined as:

Shape	Parameters	Notes
NXgeneralsphere	x,y,z,r	Centre, radius
NXgeneralcylinder	nx,ny,nz,x,y,z,h,r	Normal, face centre point, height, radius.
NXgeneralcuboid	x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4	First point is centre of 3 edges of the cuboid formed with remaining points.
NXgeneralhexahedron	x1,x2,x3,….x8,y8,z8	Points ordered anticlockwise, bottom face first
NXgeneralcone	nx,ny,nz,x,y,z,h,a	Normal, apex, height, angle

NXgeneraltorus	nx,ny,nz,x,y,z,r1,r2	Normal, centre, centre radius, shell radius.
NXinfiniteplane	nx,ny,ny,p1,p2,p3	Infinite plane
NXinfinitecylinder	nx,ny,nz,px,py,pz,r	Normal, point,radius
NXinfinitecone	nx,ny,nz,px,py,pz,a	Normal, apex, angle
NXgeneralquadraticsurface	a,b,c,d,e,f,g,h,j,k	Quadratic coefficients.

A general CSG object is built by the operations intersection, union and complement. A possible way to allow this is to define a new type of shape, NXcsgshape. Instead of being defined from a set of standard shapes it will be defined in terms of previously defined objects and the above operations. This definition can be described as shown here.

NXcsgshape csgshape

Re	Name	Attibute	Type	Value	Description
1+	{shape_name_n}		NXshape		One or more named shapes e.g. cyl1, planeX3
1	csg_string		NX_char		CSG operations on named shapes to give final shape. Default is intersection, ":" is union and "#" is complement. E.g "(cyl1 : cyl2) sph3" is union of two cylinders then intersection with a sphere.

For example a box could be defined as the intersection of six infinite planes and a cylinder with a conical end as the union of a general cylinder with a general cone.

References

1. *NeXus Instrument Definitions for ISIS Muon Data*, (revision 5, version 2), Stephen Cottrell (May 2008). A version of this is available from http://www.isis.rl.ac.uk/muons/data_analysis/nexus/intro.htm

2. NeXus NXprocessed format, http://www.nexusformat.org/Processed_Data .

3. http://mcnp-green.lanl.gov/publication/pdf/LA-UR-05-4983_Monte_Carlo_Lectures.pdf

4. NeXus TOFRaw http://www.nexusformat.org/TOFRaw

5. http://en.wikipedia.org/wiki/Constructive_solid_geometry