# NeXus data format options for Mantid

## 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 processed data out in a similar file format. NeXus is a general format for describing X-ray and neutron 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 draft DTDs for Time of Flight (TOF) RAW data, Muon data and processed data. These DTDs build on the available fields defined by NeXus to specify the data which is required or optional for a given area.

## NXprocessed

The NXprocessed DTD represents a basic format that could be used to store some types of Mantid Workspace data. It is intended to represent a set of data that has been processed so that it no longer requires the instrument definition to interpret it. Hence it would not be a complete format for a workspace that had just been read in from a RAW file or from the DAE, or for one that has only been partially processed. However it could still serve to store the heart of the workspace data, storing the partially processed data. It could be written alongside a copy of other sections, such as NXinstrument, if this is still required.

NXprocessed is described by the following DTD:

<!--

URL: http://www.nexusformat.org/classes/xml/NXprocessed.xml

Editor: NIAC

NIAC Version: 0.1

$Id$

Template of a generic NXentry containing processed data.

The assumption is that measured data, which could, for example, stored in another

NXentry within the same file, has been reduced to a standard form, e.g., S(Q), so

that the instrument information is no longer required. It is only necessary, therefore,

to store the multidimensional array containing the processed data within one or more

NXdata groups.

This is not a true metaDTD because both the values and the axis names can use

something more descriptive.

-->

<NXentry name="{Name of entry}">

<title>

{Extended title for entry}

</title>

<definition URL="http://www.nexusformat.org/instruments/xml/NXprocessed.xml"

version="1.0">

NXprocessed

</definition>

<NXsample name="{Name of sample}">?

{Any relevant sample information necessary to define the data.}

</NXsample>

<NXdata name="{Name of processed data}">

<values signal="1" type="NX\_FLOAT[:,:]" axes="axis1:axis2">{Processed values}</values>

<axis1 type="NX\_FLOAT[:]">{Values of the first dimension's axis}</axis1>

<axis2 type="NX\_FLOAT[:]">{Values of the second dimension's axis}</axis2>

</NXdata>

<NXprocess name="{Name of process}">?

{Any relevant information about the steps used to process the data.}

</NXprocess>

</NXentry>

The details of the NXsample and NXprocess are left open to the requirements of the implementers. Detailed NXsample definitions are available for other DTD’s which are appropriate to the experiments they are describing. A list of standard fields is available on the Nexus website at <http://www.nexusformat.org/NXsample>. The field NXsample can also contain other NeXus fields, such as NXgeometry, NXbeam, NXenvironment, etc. to describe aspects of the sample.

### NXsample data

The NXsample fields written by Mantid would the same as was read from the input file if this was a Nexus source. There is some ambiguity here because the contents of the NXsample section varies between formats. For example in the original Muon Nexus file format the NXlog sections of data are written outside of the NXsample section. The more recent DTD of the Muon format requires that the sample specific log data to be written in the NXsample entry which makes more sense. However, not all NXlog data should go in to NXsample. Other classes such as NXinstrument can also have NXlog data.

For the case of reading Isis raw data files, there is no simple way to classify if log data read from one of the many possible text files, *“<filename>\_<valuename>.txt*”, should be treated as belonging to the sample section or some other section. If the set of possible *<valuenames>* is small and well defined a mapping could be made so that only appropriate data from an Isis Raw file is saved in the NXsample section.

### NXprocess data

A typical Mantid 2D workspace, generated by LoadRawNexus on the file HET15869.RAW,this currently contains the additional fields for:

* m\_axes (information on the axes including titles)
* m\_title (workspace title)
* m\_comment (comments on workspace)
* sptr\_intrument (shared pointer to instrument description)
  + MantidGeometryCompAssembly
  + \_detectorCache
  + \_sourceCache
  + \_sampleCache
* sptr\_spectramap (shared pointer to spectra to detector map)
  + \_s2dmap (24962 null pointers?)
* sptr\_sample (shared pointer to sample data)
  + m\_name
  + m\_manager
    - m\_properties
    - m\_orderedProperties (sample ordered properties, such as time series values)
      * m\_isDefault (true)
      * m\_name (Log source file name (RAW) or Log name (Muon data)
      * m\_documentation
      * m\_typeinfo
      * m\_direction (0)
* m\_history
  + m\_environment
    - OSname/version/etc
  + m\_algorithm[0]
    - for each applied algorithm name/version/date/duration/properties (set if run as a managed algorithm)
* m\_isdistribution (false)
* m\_noVectors
* data
  + X(),Y(),E()

The LoadRawTest does not populate all possible fields. The Log data is stored in the sample data under m\_manager/m\_orderedProperties.

The data fields will differ from the data read from the input file, even if this just by the addition of an error value, and needs to be stored in a new Nexus file section. The instrument data is likely to be read either as an external XML file or from such data stored in Nexus format. Below we discuss how a Nexus NXprocess entry can be used to store data from algorithm history section along with the data fields. Fields such as the axes, title and comment can also be stored in then NXprocess entity.

The instrument data is stored as XML, possibly converted into a format that can be stored within Nexus. If this is the case it can be copied into the new Nexus output file, or a link made pointing to the file name.

The properties associated with the sample will be the Log files and the other properties (e.g. sample temperature) that were read in from the original Nexus file (or other source, e.g.LoadRaw text files). In the main these can be copied through to a new NXsample section with NXprocess.

Certain properties generated by the algorithm can be properties that are not usually a workspace, for example the output of a fitting algorithm might be a set of 3-tuples giving parameter names, values and errors. There needs to be a way to store algorithm such algorithm output results in Nexus format, but for the moment we focus on the case of just saving a 2D workspace.

### Workspace storage in NXdata format

The arrays of a workspace can be stored in an NXdata section within an NXprocess entry. The format assumes that the data is two dimensional with on a grid with identical axes in the two dimensions. This can easily be adapted to saving a 1D workspace. If we allow the case of each spectra having a different set of X coordinates, each spectra will have to be written separately. An NXnote field could be used to state that each of these NXdata sections comes from a single workspace.

### Algorithm history storage in NXprocess format

To save algorithm history data in the Nexus NXprocessed format, the existing fields can be used. The NXprocess section of NXprocessed is just 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. The NXprocess definition is:

<?xml version="1.0" encoding="UTF-8"?>

<!--

URL: http://www.nexus.anl.gov/classes/xml/NXlog.xml

Editor: NIAC

$Id: NXprocess.xml 4 2005-07-19 04:10:26Z rio $

Template for a process.

-->

<NXprocess name="">

<NXnote name="{numbered name to allow for ordering steps}">

{}{The note will contain information about how the data was processed. The contents of the note can be anything that the processing code can understand, or simple text.}+

</NXnote>

</NXprocess>

The NXnote fields are:

<?xml version="1.0" encoding="UTF-8"?>

<!--

URL: http://www.nexus.anl.gov/classes/xml/NXnote.xml

Editor: NIAC

$Id: NXnote.xml 4 2005-07-19 04:10:26Z rio $

This class can be used to store additional information in a NeXus file e.g.

pictures, movies, audio, additional text logs

-->

<NXnote name="{name of note}">

<author type="NX\_CHAR">

{Author or creator of note}?

</author>

<date type="ISO8601">

{Date note created/added}?

</date>

<type type="NX\_CHAR">

{Mime content type of note data field}?

</type>

<file\_name type="NX\_CHAR">

{Name of original file if note read from external source}?

</file\_name>

<description type="NX\_CHAR">

{Title of an image or other details of the note}?

</description>

<data type="NX\_BINARY">

{Binary note data - if text, line terminator is \r\n.}?

</data>

</NXnote>

Of these fields the following might be used to describe the application of a single Mantid algorithm, for example LoadRaw:

<?xml version="1.0" encoding="UTF-8"?>

<!--

URL: http://www.nexus.anl.gov/classes/xml/NXnote.xml

Editor: NIAC

$Id: NXnote.xml 4 2005-07-19 04:10:26Z rio $

This class can be used to store additional information in a NeXus file

-->

<NXnote name="Mantid-Step=1">

<author type="NX\_CHAR">

Mantid/username

</author>

<date type="ISO8601">

20080917T12:00:00

</date>

<type type="NX\_CHAR">

text/plain

</type>

<description type="NX\_CHAR">

Mantid algorithm “Loadraw”

Read Isis raw file into workspace

Read associated log files into workspace

Read instrument description into workspace

Read detector map into workspace

</description>

<data type="NX\_BINARY">

Algorithm\_name=”loadraw”

Algorithm\_version=1.0

Algorithm\_parameters=(filename ../../Test/Data/HET15869.RAW)

(workspace outputworkspace)

</data>

</NXnote>

The same format can be used for subsequent algorithmic steps, up to the one that produces the final result. The algorithm is described within the description section in a text format so others can understand the purpose of it. In the data section the algorithm name and the required parameters are given in a set of keyword value pairs. The optional parameters take their default values unless listed.

The environment data, essentially the operating system information, should also be written out here as it may be useful to know the system that the algorithm was executed on. A separate NXnote will be used for this.

For example:

<NXnote name="Mantid-environment">

<date type="ISO8601">

20080917T12:00:00

</date>

<type type="NX\_CHAR">

text/plain

</type>

<description type="NX\_CHAR">

Mantid environment

</description>

<data type="NX\_BINARY">

Version=1

Osname=Windows NT

Osversion=5.1

Username=rff93

</data>

</NXnote>

For the case of multiple algorithms applied to one or more data sets the m\_history field contains the sum of histories for each input workspace. For example, if we take the LoadNexus algorithm to create a workspace and then use the Plus algorithm to add two copies of this workspace together, we get three entries in the m\_history/m\_algorithm section, the first two being “LoadNexus” and the third “Plus”. It is not likely that two identical spectra would be added together in practice, though it is possible that a background might be fitted to (parts) of the original data and then subtracted from the original. It is not a problem to have the same entries repeated in the history and it is needed to show the data flow.

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. Other software packages could use a similar format to describe the algorithms they used, but these will also only be fully understood with the package being used.

## NXsample data

The NXsample data is set by what can be read from the input file. In the case of the example EMU files it contains just the name, temperature and magnetic field data, along with a number of time series properties. These can be written to the new data file. If the processed data is stored in the same file as the original data it would make more sense to create a link to the original NXsample section rather than making a copy. The user may create additional data about the sample during processing – for example adding unit cell parameters which were not in the original NXsample data. Additional values could be stored in Mantid as properties (or time series properties) in the workspace->sample log. These could then be written out into the new NXsample section.

One problem that may occur in mapping data from a Mantid workspace->sample area to a Nexus file NXsample section is the fact that differing proposals for NXsample suggest changing or adding field names to NXsample. As the standard evolves this should become less of a problem. At present an algorithm is free to add any name and property value to the workspace->sample. If Nexus is to be used as the format to save data in it makes sense to use the Nexus approved names for these values wherever possible.

For data integrity it is important that users can record any changes they have to make to the information, such as substituting a better value of the unit cell size. So if an algorithm allows changing data that is saved out in the new NXsample section it should record the change.