Design

Table of Contents

NeXus Objects 1
Data Items 1
Data Attributes 1
Data Groups 1
NeXus Classes 2
NeXus Data 4
NeXus Attributes 6

The structure of NeXus files is extremely flexible, allowing the storage both of simple data sets, e.g., a single data array and its axes, and also of highly complex data, e.g., the simulation results of an entire multi-component instrument. This flexibility is achieved through a hierarchical structure, with related data items collected together into groups, making NeXus files easy to navigate, even without any documentation. NeXus files are self-describing, and should be easy to understand, at least by those familiar with the experimental technique.

The logical design is distinct from the underlying format used to store the NeXus file on disk, which are written using the NeXus Application Program Interface (API). Refer to the API section for more details.

NeXus Objects

NeXus data files contain two types of elementary object: data items and data groups. In addition, metadata required to describe a data item, e.g. its physical units, can be attached to the data as data attributes.

Data Items

Data items contain the essential information stored in a NeXus file. They can be scalar values or multidimensional arrays of a variety of sizes (1-byte, 2-byte, 4-byte, 8-byte) and types (integers, floats, characters). The items may store both experimental results (counts, detector angles, etc), and other information associated with the experiment (start and end times, user names, etc). Data items are identified by their names, which must be unique within the group in which they are stored.

Data Attributes

Attributes are extra (meta-)information that are associated with particular data items. They are used to annotate the data, e.g. with physical units or calibration offsets, and may be scalar numbers or character strings. In addition, NeXus uses attributes to identify plottable data and their axes, etc. Finally, NeXus files themselves have global attributes that identify the NeXus version, file creation time, etc.. Attributes are identified by their names, which must be unique in each data item.

Data Groups

NeXus files consist of data groups, which contain data items and/or other groups to form a hierarchical structure. This hierarchy is designed to make it easy to navigate a NeXus file by storing related data items together. Data groups are identified both by a name, which must be unique within a particular group, and a class. There can be multiple groups with the same class.

NeXus Classes

Data groups often describe objects in the experiment (monitors, detectors, monochromators, etc.), so that the contents (both data items and/or other data groups) comprise the properties of that object. NeXus has defined a set of standard objects, or classes, out of which a NeXus file can be constructed. Each data group is therefore identified by a name and a class. The group class, which always has "NX" as a prefix, defines the type of object and the properties that it can contain, whereas the group name defines a unique instance of that class. These classes are defined in XML using the NeXus MetaDTD format.

Not all classes define physical objects. Some refer to logical groupings of experimental information, such as plottable data, sample environment logs, beam profiles, etc.

The following table shows the hierarchy of a standard NeXus file, and where groups of a particular class are located. There can be multiple instances of each class. On the other hand, a typical NeXus file will only contain a small subset of the possible classes.

Click on any of the class names below for a more detailed description compiled from the latest XML files.

Table 1. Design

NXroot

The root level of a NeXus file

NXentry

All the data, including instrument and sample descriptions, which logically make up a single scan or measurement. At many facilities, this corresponds to the entity that is defined by a single run number, which could be used to name the NXentry group. There can be many NXentry groups in each NeXus file

NXinstrument

The information needed to describe the instrument. This group contains other groups that describe instrument components e.g. choppers, collimators, detectors.

NXaperture

NXattenuator

NXbeam_stop

NXbending_magnet

NXcollimator

NXcrystal

NXdetector

NXdisk_chopper

NXfermi_chopper

NXfilter

NXflipper

NXguide

NXinsertion_device

NXmirror

NXmoderator

NXmonochromator

NXpolarizer

NXpositioner

NXsource

NXvelocity_selector

Table 2. Design continued

NXsample

The information needed to define the physical state of the sample during the scan *e.g.* temperature, magnetic field, crystal mosaic.

NXmonitor

Monitor data, i.e., counts, integrals, *etc*.

NXdata

The data to be plotted i.e. a single data set comprising the measurements along with the data errors, and the default axis scales and labels required to plot the data. There can be more than one NXentry, e.g., if there are several detector banks producing plottable data.

NXevent_data

Event-based data, i.e. a data set in which each count is recorded as a separate data event. This form might be chosen when the data are too sparse to be stored efficiently in histograms. Normally, such data will have to be converted to a regular NXdata group before it can be analyzed.

NXuser

Details of a user, i.e., name, affiliation, email address, *etc*.

NXprocess

Group used to store details of how the data have been processed.

NXcharacterization

Information required for the analysis of this NXentry, *e.g.* identification of empty can runs, vanadium runs, *etc.*

In addition to these classes, which appear in the locations shown above, the following groups can be added to any group in a NeXus file.

NeXus Data

One of the aims of the NeXus design was to make it possible to separate the measured data in a NeXus file from all the metadata that describe how that measurement was performed. In principle, it should be possible for a plotting utility to identify the plottable data automatically (or to provide a list of choices if there is more than one set of data). In order to distinguish the actual measurements from this metadata, it is stored separately in groups with the class NXdata. These groups encapsulate all the information required to produce a meaningful plot, including any error arrays and axis scales, i.e. the physical values corresponding to the data dimensions.

The NXdata groups have to be flexible enough to cope with data of arbitrary rank and provide a mechanism for associating axis scales with the appropriate dimension of data. We use data attributes to accomplish this. Here are the main rules that must be followed in constructing an NXdata group.

- Each NXdata group will consist of only one data set containing plottable data and their standard deviations.
- The data set will be identified by an attribute of "signal" given a value 1.
- This data set may be of arbitrary rank.
- For each data dimension, there should be a one-dimensional array of the same length.
- These one-dimensional arrays are the "dimension scales" of the data i.e. the values of the independent variables at which the data is measured e.g. scattering angle or energy transfer.

There are two methods of linking each data dimension to its respective dimension scale.

1. The first method is to define an attribute of each dimension scale called "axis". It is an integer whose value is the number of the dimension, in order of fastest varying dimension. i.e. if the array

being stored is data, with elements `data[j][i]` in C and `data(i,j)` in Fortran, where `i` is the time-of-flight index and `j` is the polar angle index, the NXdata group would contain :

```
<NXdata name=" data ">
    <time_of_flight axis= 1 primary= 1>
        1500.0 1502.0 1504.0 …
    </time_of_flight>
    <polar_angle axis= 2 primary= 1>
        15.0 15.6 16.2 …
    </polar_angle>
    <data> 5 7 14 … </data>
</NXdata>
```

2. The second method is to define an attribute of the data itself called "axes". It contains the names of each dimensions scale as a comma- or colon-delimited list in the order they appear in C. Optionally, the list can be enclosed in brackets, but should not contain any spaces, e.g.

```
<NXdata name=" data " >
    <time_of_flight > 1500.0 1502.0 1504.0 … </time_of_flight>
    <polar_angle > 15.0 15.6 16.2 … </polar_angle>
    <data axes="polar_angle:time_of_flight" > 5 7 14 … </data>
</NXdata>
```

The first method was historically the first to be used, but the second is now recommended for future applications. However, both will be supported in NeXus utilities that identify dimension scales, e.g. NXUfindaxis.

There are limited circumstances in which more than one dimension scale for the same data dimension can be included in the same NXdata group. The most common is when they are the three components of an (hkl) scan. In order to handle this case, we have defined another attribute of type integer called "primary" whose value determines the order in which the scale is expected to be chosen for plotting, i.e. 1st choice: primary = 1 2nd choice: primary = 2 etc. If there is more than one scale with the same value of the "axis" attribute, one of them must have the "primary" attribute set to 1. Defining the "primary" attribute for the other scales is optional.

It is often (usually) necessary to associate the data and/or axis scales with other metadata stored in other groups, e.g. the NXsample group or components of the NXinstrument group. For example, it may be necessary to perform corrections for the detector efficiency using information stored in the associated NXdetector group. In this case, it is recommended that the relevant arrays are initially stored in those groups, and then linked to the NXdata group. The API will provide a mechanism for identifying the parent group so that the relevant metadata can be accessed.

Here is a simple example to illustrate the concept:

```
<NXentry name="entry">
    <NXsample name="sample">
        <magnetic_field link="/entry/sample">10.0</magnetic_field>
    </NXsample>
    <NXinstrument name="instrument">
        <NXdetector name="detector">
            <data axes=" time_of_flight:magnetic_field "
                link="/entry/instrument/detector">5 7 14 etc </data>
            <time_of_flight link="/entry/instrument/detector ">
                1500.0 1502.0 1504.0 etc </time_of_flight>
```

```
        </NXdetector>
    </NXinstrument>
    <NXdata>
        <data axes="time_of_flight:magnetic_field"
            link="/entry/instrument/detector">
            {link to values in NXdetector}</data>
        <time_of_flight link="/entry/instrument/detector">
            {link to values in NXdetector}</time_of_flight>
        <magnetic_field link="/entry/sample">
            {link to values in NXsample}</magnetic_field>
    </NXdata>
</NXentry>
```

The general principle is that physical quantites are stored in the groups that they refer to (e.g. counts in NXdetector, temperature in NXsample) and these quantities are then linked into NXdata for interpretation. In this example, there are two axis scales, "magnetic_field" and "time_of_flight", which are stored in NXsample and NXdetector groups respectively. A program is able to use the information in the "link" attribute to locate the respective groups. One corollary of this is that there should be one NXdetector group for each NXdata group, e.g. one for each detector bank in a multi-bank instrument.

The syntax of the "link" attribute requires a bit of explanation. Under HDF4 you can only create, what would be called under UNIX, "hard links". Hard links have the characteristics that:

- The name of the entity must be the same in both the original and linked groups
- The attributes of both the origial entity and the linked one are the same
- You cannot distinguish the original entity from the linked one
- You cannot follow a link - it is like an inode in a filesystem and just points at the data

To overcome this and allow us to link from NXdata to, say, NXsample and to know that the original data belongs to NXsample we write the "link" attribute that contains the path of the original group containing it. All linked entities will share this "link" attribute and thus can use it to locate the original source group. We are effectively using the "link" attribute to simulate "symbolic links". So in the above example both the original "time_of_flight" and the linked one will share a link attribute containing the text "/entry/instrument/detector" because "/entry/instrument/detector/time_of_flight" is the original instance.

NeXus Attributes

Table 3. Global Attributes

Name	Type	Description
file_name	NX_CHAR	File name of original NeXus file to assist in identification if the external name has been changed
file_time	ISO 8601	Date and time of file creation
file_update_time	ISO 8601	Date and time of last file change at close
NeXus_version	NX_CHAR	Version of NeXus API used in writing the file
creator	NX_CHAR	Facility or program where the file originated

Table 4. Data Attributes

Name	Type	Description
units	NX_CHAR	Units of data which must conform to the standard defined by the Unidata UDunits utility (in particular, see udunits.dat)
signal	NX_INT32	Defines which data set contains the signal to be plotted - set to 1 for main signal
axes	NX_CHAR	Defines the names of the dimension scales for this data set as a comma-delimited array, optionally surrounded by brackets (see a longer discussion in the section on NXdata structure) *i.e.* if the array being stored is `data`, with elements `data[j][i]` in C and `data(i,j)` in Fortran, with dimension scales `time_of_flight[i]` and `polar_angle[j]`, `data` would have an attribute called "axes" with the following value : `[polar_angle,time_of_flight]`
axis	NX_INT32	As an alternative to using the "axes" attribute, this defines the rank of the signal data for which this data set is a dimension scale in order of the fastest varying index (see a longer discussion in the section on NXdata structure) *i.e.* if the array being stored is `data`, with elements `data[j][i]` in C and `data(i,j)` in Fortran, "axis" would have the following values :
primary	NX_INT32	Defines the order of preference for dimension scales which apply to the same rank of signal data - set to 1 for preferred dimension scale
long_name	NX_CHAR	Defines title of signal data or axis label of dimension scale
calibration_status	NX_CHAR	Defines status of data value - set to "Nominal" or "Measured"
histogram_offset	NX_FLOAT32	Defines the offset from the first data point to its bin boundary. *i.e.* left_bin = data[1] - histogram_offset - set to 0 if the data are not histograms. The points themselves should be set to the bin centers. For reasoning behind this design, see note on histograms.
checksum	NX_INT32	Sum of data array acting as a check on data integrity
version	NX_CHAR	Version of XML DTD file or schema on which the NeXus file is based. Should only be used with the "analysis" data item in an NXentry group.
URL	NX_CHAR	The URL of the XML DTD file or schema on which the NeXus file is based. Should only be used with the "analysis" data item in an NXentry group.