NeXus Manual

COLLABORATORS

	TITLE : NeXus Manual		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Ray Osborn, Mark Koennecke, Przemek Klosowski, Frederick Akeroyd, Peter F. Peterson, and Pete R. Jemian	February 6, 2010	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
	2009	Started conversion from the old NeXus mediawiki documentation.	PFP
draft	2010	Most of the content from the old NeXus mediawiki documentation is included. The manual needs significant editing before ready to release. Parts of the manual are quite rough while others are in good shape. The class descriptions are autogenerated from the NXDL source files.	PRJ

Contents

List of Figures

List of Tables

Preface

This documentation is in transition from the NeXus wiki site to DocBook. Simultaneously, NeXus is introducing the NeXus Definition Language (NXDL) to define base classes and application definitions. Parts of the manual may still need revision to conform to the NXDL.

As part of this transition of the documentation from Mediawiki to DocBook, additional examples have been added to respond to inquiry from the users of the NeXus standard about implementation and usage. Hopefully these examples and the new NXDL will reduce the learning barriers incurred by those new to NeXus.

Chapter 1

NeXus Introduction

In recent years, a number of scientists and computer programmers working in neutron and synchrotron facilities around the world came to the conclusion that a common data format would fulfill a valuable function in the scattering community. As instrumentation becomes more complex and data visualization become more challenging, individual scientists, or even institutions, have found it difficult to keep up with new developments. A common data format makes it easier, both to exchange experimental results and to exchange ideas about how to analyze them. It promotes greater cooperation in software development and stimulate the design of more sophisticated visualization tools. For additional background information see Appendix E.

This section is designed to give a brief introduction to NeXus, the data format that has been developed in response to these needs. It explains what a modern data format such as NeXus is and how to write simple programs to read and write NeXus files.

The programmers who produce intermediate files for storing analyzed data should agree on simple interchange rules .

1.1 Motivations for the NeXus standard

By the early 1990s, several groups of scientists in the fields of neutron and X-ray science had recognized a common and troublesome pattern in the data acquired at various scientific instruments and user facilities. Each of these instruments and facilities had a locally defined format for recording experimental data. With lots of different formats, much of the scientists' time was being wasted in the task of writing import readers for processing and analysis programs. As is common, the exact information to be documented from each instrument in a data file evolves, such as the implementation of new high-throughput detectors. Many of these formats lacked the generality to extend to the new data to be stored, thus another new format was devised. In such environments, the documentation of each generation of data format is often lacking.

Three parallel developments have led to NeXus:

1. *June 1994*: Mark Koennecke (Paul Scherer Institute, Switzerland) made a proposal using netCDF for the European neutron scattering community while working at the ISIS pulsed neutron facility.

2. *August 1994*: Jon Tischler and Mitch Nelson (Oak Ridge National Laboratory, USA) proposed an HDF-based format as a standard for data storage at the Advanced Photon Source (Argonne National Laboratory, USA).
3. *October 1996*: Przemek Klosowski (National Institute of Standards and Technology, USA) produced a first draft of the NeXus proposal drawing on ideas from both sources.

These scientists proposed methods to store data using a self-describing, extensible format that was already in broad use in other scientific disciplines. Their proposals formed the basis for the current design of the NeXus standard which was developed at two workshops, SoftNeSS'95 (NIST Sept. 1995) and SoftNeSS'96 (Argonne Oct. 1996), attended by representatives of a range of neutron and x-ray facilities. The NeXus API was released in late 1997. Basic motivations for this standard were:

1. Simple plotting
2. A unified format for reduction and analysis
3. A defined dictionary of terms

1.1.1 Simple plotting

An important motivation for the design of NeXus was to simplify the creation of a default plot view. While the best representation of a set of observations will vary, depending on various conditions, a good suggestion is often known *a priori*. This suggestion is described in the `NXdata` element so that any program that is used to browse NeXus data files can provide a *best representation* without request for user input.

1.1.2 Unified format for reduction and analysis

Another important motivation for NeXus, indeed the *raison d'etre*, was the community need to analyze data from different user facilities. A single data format that is in use at a variety of facilities would provide a major benefit to the scientific community. This unified format should be capable of describing any type of data from the scientific experiments, at any step of the process from data acquisition to data reduction and analysis. By the late 1980s, it had become common practice for a scientific instrument or facility to define its own data format, often at the convenience of the local computer system. Data from these facilities were not easily interchanged due to various differences in computer systems and the compression schemes of binary data. It was necessary to contact the facility to obtain a description so that one could write an import routine in software. Experience with facilities closing (and subsequent lack of access to information describing the facility data format) revealed a significant limitation with this common practice.

Self-description, combined with a reliance on a **multi-platform** (and thereby **portable**) data storage format, were valued components of a data storage format where the longevity of the data was expected to be longer than the lifetime of the facility at which it was acquired. As the name implies, self-description within data files is the practice where the structure of the information contained within the file is evident from the file itself. A multi-platform data storage format must faithfully represent the data identically on a variety of computer systems, regardless of the bit order or byte order or word size native to the computer.

The scientific community continues to grow the various types of data to be expressed in data files. This practice is expected to continue as part of the investigative process. To gain broad acceptance in the scientific user community, any data storage format proposed as a standard would need to be **extendable** and continue to provide a means to express the latest notions of scientific data.

The maintenance cost of common data structures meeting the motivations above (self-describing, portable, and extendable) is not insurmountable but is often well-beyond the research funding of individual members of the muon, neutron, and X-ray science communities. Since it is these members that drive the selection of a data storage format, it is necessary for the user cost to be as minimal as possible. In this case, experience has shown that the format must be in the **public-domain** for it to be commonly accepted as a standard. A benefit of the public-domain aspect is that the source code for the API is open and accessible, a point which has received notable comment in the scientific literature.

At its beginnings, the founders of NeXus identified the Hierarchical Data Format (HDF), initially from the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign (UIUC) and later spun off into its own group called The HDF Group (THG),[1] as a multi-platform data storage format with capacity for conveying large data payloads

[1] The HDF Group: http://www.hdfgroup.org/

and a substantial user community. HDF (now HDF5) was provided with software to read and write data (this is the application-programmer interface, or API) using a large number of computing systems in common use for neutron and X-ray science. HDF is a binary data file format that supports compression and structured data.

More recently, NeXus has recognized that part of the scientific community with a desire to write and record scientific data, has small data volumes and a large aversion to the requirement of a complicated API necessary to access data in binary files such as HDF. For such information, the NeXus API has been extended by the addition of the eXtensible Markup Language[2] (XML) as an alternative to HDF. XML is a text-based format that supports compression and structured data and has broad usage in business and e-commerce. While possibly complicated, XML files are human readable, and tools for translation and extraction are plentiful. The API has routines to read and write XML data and to convert between HDF and XML.

1.1.3 Defined dictionary of terms

A necessary feature of a standard for the interchange of scientific data is a *defined dictionary* (or *lexicography*) of terms. This dictionary declares the expected spelling and meaning of terms when they are present so that it is not necessary to search for all the variant forms of *energy* when it is used to describe data (e.g., `E`, `e`, `keV`, `eV`, `nrg`, ...).

NeXus recognized that each scientific specialty has developed a unique dictionary and needs to categorize data using those terms. The NeXus Application Definitions provide the means to document the lexicography for use in data files of that scientific specialty.

1.2 What is NeXus?

The NeXus data format has four components:

A set of subroutines (utilities) to make it easy to read and write NeXus files.

A set of design principles to help people understand what is in them.

A set of data storage objects (base classes and application definitions) to allow the development of more portable analysis software.

A set of low-level file formats to actually store NeXus files on physical media.

Each of these components is described in this manual. Section 1.4 describes the physical file format of NeXus data files. The NeXus Application-Programmer Interface (NAPI), which provides the set of subroutines for reading and writing NeXus data files, is described briefly in Section 1.5 while more details are provided in Chapter 3. The principles guiding the design and implementation of the NeXus standard are described in Chapter 2 . Appendix A describes the definitions of base classes and applications which comprise the data storage objects used in NeXus data files. Section 1.2.4 describes the implementation of NeXus in each of the low-level formats (HDF and XML). Additionally, a brief list describing the set of NeXus Utilities available to browse, validate, translate, and visualise NeXus data files is provided in Chapter B.

1.2.1 A Set of Subroutines

In the past, a data format was defined by a document describing the precise location of every item in the data file, either as row and column numbers in an ASCII file, or as record and byte numbers in a binary file. In modern data formats, such as NeXus, the user does not need to be concerned where the data are stored, just what they are called. It is the job of the subroutine library to retrieve the data. This subroutine library is commonly called an application-programmer interface or API.

For example, in NeXus, a program to read in the wavelength of an experiment would contain lines similar to the following:

Example 1.1 Simple example of reading data

```
NXopendata (fileID, "wavelength");
NXgetdata (fileID, lambda);
NXclosedata (fileID);
```

[2] XML: http://www.w3.org/XML/. There are many other descriptions of XML, for example: http://en.wikipedia.org/wiki/XML

In this example, the program requests the value of the data that has the label `wavelength`, storing the result in the variable lambda. `fileID` is a file identifier that is provided by NeXus when the file is opened.

We shall provide a more complete example when we have discussed the contents of the NeXus files.

1.2.2 A Set of Design Principles

NeXus data files contain three types of entity: data groups, data fields, and attributes.

Data Groups Data groups are like folders that can contain a number of fields and/or other groups.

Data Fields Data fields can be scalar values or multidimensional arrays of a variety of sizes (1-byte, 2-byte, 4-byte, 8-byte) and types (characters, integers, floats).

Data Attributes Extra information required to describe a particular group or field, such as the data units, can be stored as a data attribute.

In fact, a NeXus file can be viewed as a computer file system. Just as files are stored in folders (or subdirectories) to make them easy to locate, so NeXus fields are stored in groups. The group hierarchy is designed to make it easy to navigate a NeXus file.

1.2.2.1 Example of a NeXus File

The following diagram shows an example of a NeXus file represented as a tree structure.

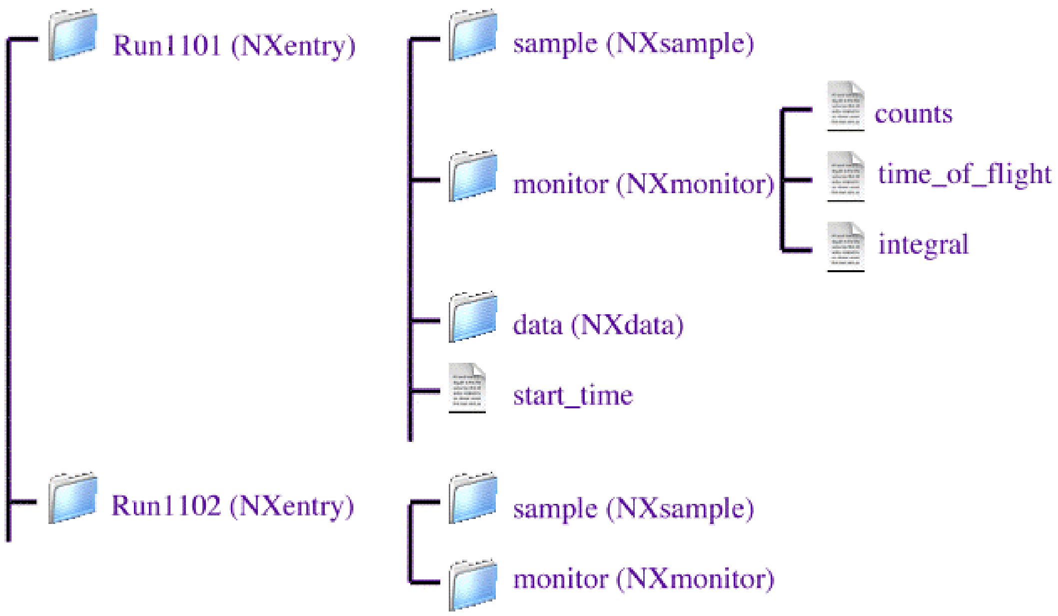

Figure 1.1: Example of a NeXus file

Note that each field is identified by a name, such as `counts`, but each group is identified both by a name and, in parentheses, a class identifier, e.g., monitor (`NXmonitor`). The class names, which all begin with `NX`, define the sort of fields that the group should contain, in this case, counts from a beamline monitor. The hierarchical design, with data items nested in groups, makes it easy to identify information if you are browsing through a file.

1.2.2.2 Important Classes

Here are some of the important classes found in nearly all NeXus files. A complete list can be found in the NeXus Design section

NXentry The top level of any NeXus file contains one or more groups with the class `NXentry`. These contain all the data that is required to describe an experimental run or scan. Each `NXentry` typically contains a number of groups describing sample information (class `NXsample`), instrument details (class `NXinstrument`), and monitor counts (class `NXmonitor`).

NXdata Each `NXentry` group contains one or more groups with class `NXdata`. These groups contain the experimental results in a self-contained way, i.e., it should be possible to generate a sensible plot of the data from the information contained in each `NXdata` group. That means it should contain the axis labels and titles as well as the data.

NXsample A `NXentry` group will often contain a group with class `NXsample`. This group contains information pertaining to the sample, such as its chemical composition, mass, and environment variables (temperature, pressure, magnetic field, etc.).

NXinstrument There might also be a group with class `NXinstrument`. This is designed to encapsulate all the instrumental information that might be relevant to a measurement, such as flight paths, collimations, chopper frequencies, etc.

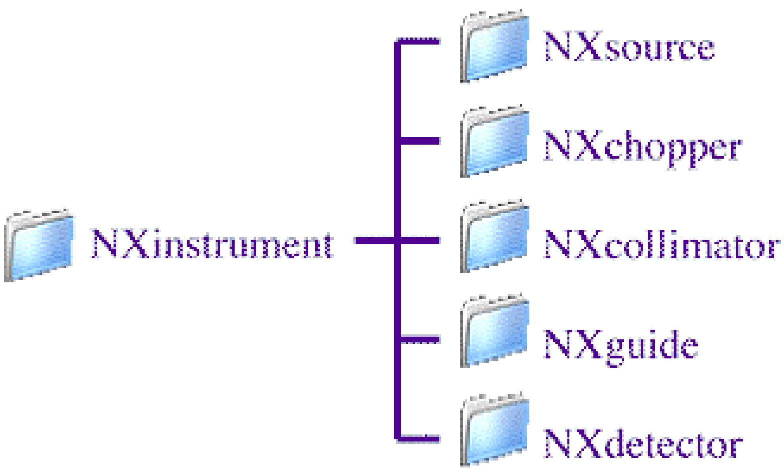

Figure 1.2: NXinstrument excerpt

Since an instrument can comprise several beamline components each defined by several parameters, they are each specified by a separate group. This hides the complexity from generic file browsers, but makes the information available in an intuitively obvious way if it is required.

1.2.2.3 Simple Example

NeXus data files do not need to be complicated. In fact, the following diagram shows an extremely simple NeXus file that could be used to transfer data between programs.

Figure 1.3: Simple Example

This illustrates the fact that the structure of NeXus files is extremely flexible. It can accommodate very complex instrumental information, if required, but it can also be used to store very simple data sets.

1.2.3 A Set of Data Storage Objects

If the design principles are followed, it will be easy for anyone browsing a NeXus file to understand what it contains, without any prior information. However, if you are writing visualization or analysis software, you will need to know precisely what information is contained in advance. For that reason, NeXus provides a way of defining the format for particular instrument types, such as time-of-flight small angle neutron scattering. This requires some agreement by the relevant communities, but enables the development of much more portable software.

These instrument definitions are being formalized as XML files, using a specially devised syntax that specifies the names of data fields, and whether they are optional or required. The following is an example of such a file for the simple NeXus file shown above.

Example 1.2 *verysimple.xml*: A very simple NeXus file

```
<?xml version="1.0" ?>
<!--
URL: http://www.neutron.anl.gov/nexus/xml/simple.xml
Editor: Ray Osborn <ROsborn@anl.gov>

A very simple NeXus file

-->
<NXentry name="{Name of entry}">
   <NXdata name="{Name of data}">
     <time_of_flight units="microseconds" type="NX_FLOAT32[i]">{Time-of-flight}</time_of_flight>
     <data type="NX_INT32[i]" axes="time_of_flight"> {Counts} </data>
   </NXdata>
</NXentry>
```

If you want to define the format of a particular type of NeXus file for your own use, e.g. as the standard output from a program, you are encouraged to "publish" the format using this XML format.

1.2.4 A Set of Low-Level File Formats

To actually store NeXus files on physical media, different low-level file formats are available, namely HDF4, HDF5, and XML. The NeXus code library may be configured to support all of them, or any nonempty subset. Applications that create NeXus files need to decide (or let the user decide) in which low-level format data shall be stored. Generic data analysis applications should be able to read any low-level format.

Give details on the HDF files and the XML files. Note that there is no root element in HDF files while NXroot is the root element in XML files. Note about the implementation of groups, fields, and attributes in HDF and XML files. Compare and contrast them.

1.3 Generic features of NeXus files

1.3.1 Groups, Fields, and Attributes

This is a paragraph.

1.3.2 Links

This is a paragraph.

1.3.2.1 Linking data to detectors

This is a paragraph.

1.3.3 Monitors as a special kind of data

This is a paragraph.

1.4 Physical File format

Describe implementation of the NeXus standard in HDF4, HDF5, and XML formats.
For example, note how groups are named in HDF and XML.
Compare and contrast the two formats (no root element in HDF).
Perhaps advise when to use one or the other.

1.5 The NeXus Application Programming Interface (API)

The NeXus API consists of routines to read and write NeXus data files.

1.5.1 How do I write a NeXus file?

The NeXus Application Program Interface (API) provides a set of subroutines that make it easy to read and write NeXus files. These subroutines are available in C, Fortran 77, Fortran 90, Java, and IDL. Access from other languages, such as Python, is anticipated in the near future. It is also possible to read NeXus files in a number of data analysis tools, such as LAMP, ISAW, and Open GENIE.

The API uses a very simple "state" model to navigate through a NeXus file. When you open a file, the API provides a file "handle", which then stores the current location, i.e. which group and/or field is currently open. Read and write operations then act on the currently open entity. In the following, we walk through some parts of a typical NeXus program written in C. See the NeXus API chapter for a more complete version.

First, it is necessary to open the file, specifying whether we want read or write access.

Example 1.3 Open a file

```
#include "napi.h"

 int main()
 {
   NXhandle fileID;
   NXopen ('NXfile.nxs', NXACC_CREATE, &fileID);
```

The file is opened with *create* access (implying write access), and the API returns a file identifier of type `NXhandle`. Next, we create an `NXentry` group to contain the scan using `NXmakegroup()` and then open it for access using `NXopengroup()`.

Example 1.4 Create an entry

```
   NXmakegroup (fileID, "entry", "NXentry");
   NXopengroup (fileID, "entry", "NXentry");
```

The plottable data is contained within an `NXdata` group, which must also be created and opened.

Example 1.5 Create data group

```
   NXmakegroup (fileID, "data", "NXdata");
   NXopengroup (fileID, "data", "NXdata");
```

To create a field, call `NXmakedata()`, specifying the data name, type (`NX_FLOAT32`), rank (in this case, `1`), and length of the array (`n_t`). Then, it can be opened for writing.

Example 1.6 Create data array

```
   NXmakedata (fileID, "time_of_flight", NX_FLOAT32, 1, &n_t);
   NXopendata (fileID, "time_of_flight");
```

Then write the data using `NXputdata()`.

Example 1.7 Write the data

```
  NXputdata (fileID, t);
```

With the field is still open, we can also add some data attributes, such as the data units, which are specified as a character string (type `NX_CHAR`) that is 12 bytes long.

Example 1.8 Add an attribute

```
NXputattr (fileID, "units", "microseconds", 12, NX_CHAR);
```

Then we close the field before opening another. In fact, the API will do this automatically if you attempt to open another field, but it is better style to close it yourself.

Example 1.9 Close the data array

```
NXclosedata (fileID);
```

The remaining fields in this group are added in a similar fashion. Note that the indentation whenever a new field or group are opened is just intended to make the structure of the NeXus file more transparent.

Example 1.10 The rest of the data group

```
    NXmakedata (fileID, "polar_angle", NX_FLOAT32, 1, &n_p);
       NXopendata (fileID, "polar_angle");
         NXputdata (fileID, polar_angle);
         NXputattr (fileID, "units", "degrees", 7, NX_CHAR);
       NXclosedata (fileID);
       dims[0] = n_t;
       dims[1] = n_p;
       NXmakedata (fileID, "counts", NX_INT32, 2, dims);
       NXopendata (fileID, "counts");
         NXputdata (fileID, counts);
       NXclosedata (fileID);
```

Finally, close the groups (`NXdata` and `NXentry`) before closing the file itself.

Example 1.11 Cleanup

```
    NXclosegroup (fileID);
    NXclosegroup (fileID);
    NXclose (&fileID);
    return;
}
```

1.5.2 How do I read a NeXus file?

Reading a NeXus file is almost identical to writing one. Obviously, it is not necessary to call `NXmakedata()` since the item already exists, but it is necessary to call one of the query routines to find out the rank and length of the data before allocating an array to store it.

Here is part of a program to read the time-of-flight array from the file created by the example above.

Example 1.12 File read example

```
NXopen ('NXfile.nxs', NXACC_READ, &fileID);
  NXopengroup (fileID, "entry", "NXentry");
    NXopengroup (fileID, "data", "NXdata");
      NXopendata (fileID, "time_of_flight");
        NXgetinfo (fileID, &rank, dims, &datatype);
        NXmalloc ((void **) &tof, rank, dims, datatype);
        NXgetdata (fileID, tof);
      NXclosedata (fileID);
    NXclosegroup (fileID);
  NXclosegroup (fileID);
NXclose (fileID);
```

NeXus files can also be viewed by a command-line browser, `NXbrowse`, which is included with the NeXus API. The following is an example session of using `nxbrowse` to view a data file from the LRMECS spectrometer at IPNS. The following commands are used (see the `nxbrowse` web page):

Table 1.1: NXbrowse Command Description

Command	Description
dir	List the contents of the current group
open Histogram1	Open the NeXus group "Histogram1"
read title	Print the contents of the NeXus data labelled "title"
close	Close the current group
quit	Quit the browser

Example 1.13 Using NXbrowse

```
%> nxbrowse lrcs3701.nxs

NXBrowse 3.0.0. Copyright (C) 2000 R. Osborn, M. Koennecke, P. Klosowski
    NeXus_version = 1.3.3
    file_name = lrcs3701.nxs
    file_time = 2001-02-11 00:02:35-0600
    user = EAG/RO
NX> dir
  NX Group : Histogram1 (NXentry)
  NX Group : Histogram2 (NXentry)
NX> open Histogram1
NX/Histogram1> dir
  NX Data  : title[44] (NX_CHAR)
  NX Data  : analysis[7] (NX_CHAR)
  NX Data  : start_time[24] (NX_CHAR)
  NX Data  : end_time[24] (NX_CHAR)
  NX Data  : run_number (NX_INT32)
  NX Group : sample (NXsample)
  NX Group : LRMECS (NXinstrument)
  NX Group : monitor1 (NXmonitor)
  NX Group : monitor2 (NXmonitor)
  NX Group : data (NXdata)
NX/Histogram1> read title
  title[44] (NX_CHAR) = MgB2 PDOS 43.37g 8K 120meV E0@240Hz T0@120Hz
NX/Histogram1> open data
NX/Histogram1/data> dir
  NX Data  : title[44] (NX_CHAR)
  NX Data  : data[148,750] (NX_INT32)
  NX Data  : time_of_flight[751] (NX_FLOAT32)
  NX Data  : polar_angle[148] (NX_FLOAT32)
NX/Histogram1/data> read time_of_flight
  time_of_flight[751] (NX_FLOAT32) = [ 1900.000000 1902.000000 1904.000000 ...]
    units = microseconds
    long_name = Time-of-Flight [microseconds]
NX/Histogram1/data> read data
  data[148,750] (NX_INT32) = [ 1 1 0 ...]
    units = counts
    signal = 1
    long_name = Neutron Counts
    axes = polar_angle:time_of_flight
NX/Histogram1/data> close
NX/Histogram1> close
NX> quit
```

The source code provides an example of how to write a NeXus reader. The test programs included in the NeXus API may also be useful to study.

1.6 NeXus Mailing Lists

There are several mailing lists associated with NeXus.

NeXus Mailing List We invite anyone who is associated with neutron and/or X-ray synchrotron scattering and who wishes to be involved in the development and testing of the NeXus format to subscribe to this list. It is for the free discussion of all aspects of the design and operation of the NeXus format.

List Address: nexus@nexusformat.org nexus@nexusformat.org
Subscriptions: http://lists.nexusformat.org/mailman/listinfo/nexus
Archive: http://lists.nexusformat.org/pipermail/nexus

NeXus Committee Mailing List This list contains discussions of the NeXus International Advisory Committee (NIAC), which oversees the development of the NeXus data format. Its members represent many of the major neutron and synchrotron scattering sources in the world. Membership and posting to this list are confined to the committee members, but the archives are public.

List Address: nexus-committee@nexusformat.org
Subscriptions: http://lists.nexusformat.org/mailman/listinfo/nexus-committee
Archive: http://lists.nexusformat.org/pipermail/nexus-committee

NeXus Developers Mailing List This mailing list is for discussions concerning the technical development of the NeXus Application Program Interface.

List Address: nexus-developers@nexusformat.org
Subscriptions: http://lists.nexusformat.org/mailman/listinfo/nexus-developers
Archive: http://lists.nexusformat.org/pipermail/nexus-developers

NeXus Code Subversion Mailing List Members of this list will receive an email whenever a commit is made to the NeXus code repository. This list cannot be posted to - all questions should instead be sent to the NeXus Developers Mailing List.

List Address: nexus-code-svn@nexusformat.org
Subscriptions: http://lists.nexusformat.org/mailman/listinfo/nexus-code-svn
Archive: http://lists.nexusformat.org/pipermail/nexus-code-svn

NeXus Definitions Subversion Mailing List Members of this list will receive an email whenever a commit is made to the NeXus definitions repository. This list cannot be posted to - all questions should instead be sent to the NeXus Developers Mailing List.

List Address: nexus-definitions-svn@nexusformat.org
Subscriptions: http://lists.nexusformat.org/mailman/listinfo/nexus-definitions-svn
Archive: http://lists.nexusformat.org/pipermail/nexus-definitions-svn

NeXus Code Tickets Mailing List Members of this list will receive an email whenever a ticket (bug/issue/task) associated with NeXus code library development is modified on the Nexus code trac server[3] This list cannot be posted to - see the section on Issue Reporting.

List Address: nexus-code-tickets@nexusformat.org
Subscriptions: http://lists.nexusformat.org/mailman/listinfo/nexus-code-tickets
Archive: http://lists.nexusformat.org/pipermail/nexus-code-tickets

NeXus Definitions Tickets Mailing List Members of this list will receive an email whenever a ticket (bug/issue/task) associated with NeXus definitions development is modified on the Nexus definitions trac server[4] This list cannot be posted to - see the section on Issue Reporting.

List Address: nexus-definitions-tickets@nexusformat.org
Subscriptions: http://lists.nexusformat.org/mailman/listinfo/nexus-definitions-tickets
Archive: http://lists.nexusformat.org/pipermail/nexus-definitions-tickets

1.7 NeXus Subversion Repositories

NeXus NXDL class definitions (both base classes and instruments) and the NeXus code library source are held in a subversion repository. The repository is world readable and though you can browse the NeXus code library and applications or NeXus NXDL class definitions repositories directly, a better looking interface is provided by the ViewVC or TRAC browsers.

Browse the NeXus code (library and applications) repository using ViewVC or TRAC
Browse NeXus definitions (NXDL classes) repository using ViewVC or TRAC

The repository can alse be interrogated for recent updates via a query form

[3] http://trac.nexusformat.org/code/report/1
[4] http://trac.nexusformat.org/definitions/report/1

```
http://svn.nexusformat.org/viewvc/NeXusCode/trunk/?view=queryform
```

For example, show me all changes in the last month for the code (library and applications) repository

```
http://svn.nexusformat.org/viewvc/NeXusCode/trunk/?view=query&date=month&limit_changes=100
```

or Definition repository

```
http://trac.nexusformat.org/definitions/timeline?daysback=30
```

If you wish to receive an email when a change is made to the repository you should join the appropriate Mailing Lists.

Table 1.2: TRAC RSS feed

Alternatively, you can use an RSS feed to keep abreast of changes. TRAC provides a link to its RSS feed on pages with an orange *XML RSS Feed* icon at their foot such as:	

There are pages that show the subversion repository activity in a timeline format or a tabular (revision log) format.

code (library and applications) repository timeline `http://trac.nexusformat.org/code/timeline`

definitions repository timeline `http://trac.nexusformat.org/definitions/timeline`

code repository revision log `http://trac.nexusformat.org/code/log`

definitions repository revision log `http://trac.nexusformat.org/definitions/log`

1.7.1 Login

To update files in these repositories you will need to use a subversion client such as `TortoiseSVN/`[5] for Microsoft Windows or `svn` for command-line shells and also provide your NeXus Wiki username and password. Note that for subversion write access:

- If your Wiki username contains a space, write it with a space (i.e. do not replace the space with an _ as is done in WIKI URLs)
- You cannot use a *temporary password* (i.e. one that was emailed to you in response to a request). You must first log into MediaWiki with the temporary password and then go to account NeXus wiki Preferences and change the password.
- Your Wiki account must have an email address associated with it and this address must have been validated. To provide and/or validate your email address, log in and go to your account NeXus wiki Preferences. section.
- If you have login problems and have not changed your WIKI password since 20th October 2006, please go to the NeXus wiki login page and request to be emailed a new password. To synchronise TRAC/Subversion/MediaWiki required some changes to the authentication system which will have invalidated passwords set prior to that date.

Here are the URLs to access the subversion repositories as a developer:

code for library/applications

```
https://svn.nexusformat.org/code/trunk
```

definitions for NXDL classes

```
https://svn.nexusformat.org/definitions/trunk
```

[5] `http://tortoisesvn.tigris.org/`

checkout the code trunk

```
svn co --username "My WIKI Username" https://svn.nexusformat.org/code/trunk nexus_code
```

Please report any problems via the Issue Reporting system.

1.7.2 Committing Changes

As well as needing a valid account, you will not be able to check-in changes unless you indicate (in the log message attached to the commit) which current issues on the Issue Reporting system the changes either fix or refer to. This is done by enclosing special phrases in the commit message of the form:

```
command #1
command #1, #2
command #1 & #2
command #1 and #2
```

where `command` is one of the commands detailed below and `#1` means *issue number 1* on the system, etc. You can have more then one command in a message. The following commands are supported and there is more then one spelling for each command (to make this as user-friendly as possible):

`closes, fixes` The specified issue numbers are closed with the contents of this commit message being added to it.

`references, refs, addresses, re` The specified issue numbers are left in their current status, but the contents of this commit message are added to their notes.

For example, the commit message

```
Changed blah and foo to do this or that. Fixes #10 and #12, and refs #12.
```

This will close issues #10 and #12, and add a note to #12 on the Issue Reporting system. For a list of current issues, see:

Active tickets for the NeXus code library: `http://trac.nexusformat.org/code/report/1`
Active tickets for NeXus definitions: `http://trac.nexusformat.org/definitions/report/1`

1.7.3 URLs described in this section

Many Uniform Resource Locators (URLs) have been used in this section. This is a table describing them.

Subversion revision management software

`http://subversion.tigris.org/`

ViewVC versions control repository viewing software

`http://www.viewvc.org/`

TRAC issue management software

`http://trac.edgewall.org`

TortoiseSVN, Windows subversion client

`http://tortoisesvn.tigris.org/`

NeXus code (library and applications) subversion repository

`http://svn.nexusformat.org/code/`

NeXus definitions subversion repository

`http://svn.nexusformat.org/definitions/`

ViewVC view of NeXus code (library and applications) repository
    http://svn.nexusformat.org/viewvc/NeXusCode

ViewVC view of NeXus definitions repository
    http://svn.nexusformat.org/viewvc/NeXusDefinitions

TRAC view of NeXus code (library and applications) repository
    http://trac.nexusformat.org/code/browser

NeXus code (library and applications) revision log
    http://trac.nexusformat.org/code/log

Active tickets for the NeXus code repository
    http://trac.nexusformat.org/code/report/1

NeXus code repository timeline
    http://trac.nexusformat.org/code/timeline

TRAC view of NeXus definitions repository
    http://trac.nexusformat.org/definitions/browser

NeXus definitions revision log
    http://trac.nexusformat.org/definitions/log

Active tickets for NeXus definitions
    http://trac.nexusformat.org/definitions/report/1

NeXus definitions repository timeline
    http://trac.nexusformat.org/definitions/timeline

NeXus code repository (password required)
    https://svn.nexusformat.org/code/trunk

NeXus definitions repository (password required)
    https://svn.nexusformat.org/definitions/trunk

1.8 NeXus Issue Reporting

NeXus is using TRAC[6] for problem/issue reporting. You can browse issues without logging on, but to report issues you will need to login using your NeXus WIKI username and password (the subversion login notes mentioned for write access to the Subversion Server apply to TRAC login, too).

Whenever an update is made to a ticket, a message is also posted to the appropriate ticket mailing list.

1.8.1 NeXus Code (Library and Applications)

Report a new issue: http://trac.nexusformat.org/code

View current issues: http://trac.nexusformat.org/code/report/1

Archive of ticket update emails: http://lists.nexusformat.org/pipermail/nexus-code-tickets/

repository timeline (recent ticket and code changes): http://trac.nexusformat.org/code/timeline

repository roadmap: http://trac.nexusformat.org/code/roadmap

[6] http://trac.edgewall.org

1.8.2 NeXus Definitions (NXDL base classes and application definitions)

Report a new issue: http://trac.nexusformat.org/definitions

View current issues: http://trac.nexusformat.org/definitions/report/1

Archive of ticket update emails: http://lists.nexusformat.org/pipermail/nexus-definitions-tickets/

repository timeline (recent ticket and definition changes): http://trac.nexusformat.org/definitions/timeline

repository roadmap: http://trac.nexusformat.org/definitions/roadmap

Chapter 2

NeXus Design

The structure of NeXus files is extremely flexible, allowing the storage both of simple data sets, such as a single data array and its axes, and also of highly complex data, such as the simulation results of an entire multi-component instrument. This flexibility is achieved through a hierarchical structure, with related *fields*[1] 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 additional API documentation in Chapter 3 for more details.

2.1 NeXus Objects

NeXus data files contain two types of elementary object: groups and fields. In addition, metadata required to describe a group or field, such as its physical units, can be attached to the data as data attributes.

2.1.1 Data Groups

NeXus files consist of data groups, which contain fields 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 fields 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.

2.1.2 Data Fields

Data fields 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 fields may store both experimental results (counts, detector angles, etc), and other information associated with the experiment (start and end times, user names, etc). Data fields are identified by their names, which must be unique within the group in which they are stored.

2.1.3 Data Attributes

Attributes are extra (meta-)information that are associated with particular fields. 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 field.

[1] In this manual, we use the terms *field*, *data field*, and *data item* synonymously to be consistent with their meaning between NeXus data file instances and NXDL specification files.

2.2 NeXus Classes

Data groups often describe objects in the experiment (monitors, detectors, monochromators, etc.), so that the contents (both data fields and/or other data groups) comprise the properties of that object. NeXus has defined a set of standard objects, or base 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 Definition Language (NXDL) 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. 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.

2.2.1 NeXus Data

One NeXus design aim 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.

If available, the standard deviations of the data are to be stored in a data set of the same rank and dimensions, with the name "errors".

- 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. One method uses the `axis` attribute to identify with an integer the axis whose value is the number of the dimension. The second method is preferred, which uses the `axis` attribute to specify the names of each dimension's scale.

2.2.1.1 Linking by dimension number using the `axis` attribute

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:

Example 2.1 Old way of denoting axes

```
<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>
```

The `axis` attribute must be defined for each dimension scale. The `primary` attribute is unique to this method of linking.

2.2.1.2 Linking by name using the **axes** attribute

The second method is to define an attribute of the data itself called *axes*. It contains the names of each dimension's 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.

Example 2.2 Preferred way of denoting axes

```
<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>
```

2.2.1.3 Discussion of the two methods

The second method is required when the dimension scale is used in more than one `NXdata` group in a different context, e.g. it is used as the x-axis in one group and the y-axis in another.

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, such as `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 set `primary="1"`. Defining the `primary` attribute for the other scales is optional.

N.B. The `primary` attribute can only be used with the first method of defining dimension scales discussed above. In addition to the signal data, this group could contain a data set of the same rank and dimensions called `errors` containing the standard deviations of the data.

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:

Example 2.3 Abbreviated NeXus file

```
<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.

2.2.2 NeXus Attributes

Metadata (additional information that describes the primary data) is provided in NeXus data files through additional groups, fields, and attributes. Attributes are used to describe specific groups or fields, such as `name="entry"` or `axis="1"`. Global attributes, as shown in Table 2.1 apply to the entire NeXus file. In HDF files, these are global, while in XML files, these are attributes of the XML file's root element: `NXroot`. Examples of other attributes are given in Table 2.2.

Table 2.1: 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 2.2: Examples of data attributes

Name	Type	Description
units	NX_CHAR	Data units, given as character strings, must conform to the NeXus units standard. See the "NeXus units" section for details.
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	The old way of designating data for plotting, now superceded by 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: ith dimension (axis = 1), jth dimension (axis = 2), etc.
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 NXDL file on which the NeXus file is based. Should only be used with the *analysis* field in an `NXentry` group.
URL	NX_CHAR	The URL of the NXDL file on which the NeXus file is based. Should only be used with the *analysis* field in an `NXentry` group.

2.3 NeXus Coordinate System

Nexus provides two coordinate systems: a polar coordinate based system for scattering coordinates and a NXgeometry based system for physical coordinates of beamline components. The coordinate origin is discussed in Section 2.3.2.2. The usage of these two systems can be illustrated by considering a ^{3}He gas tube detector on a neutron scattering instrument:

- The polar system would describe the scattering, rather than actual, geometry. For example, the *distance* coordinate would refer to the distance from the sample to an effective measurement point within the gas tube, which would depend on neutron energy; lower energy neutrons would tend to penetrate a smaller distance within the tube, and so have a shorter secondary flight path.

- The NXgeometry system represents true spatial location and would describe a cylinder at a certain distance from the sample that never changes from one run to another

In NeXus, we never came down to defining rotation directions. The triple polar_angle, azimuthal_angle and distance define a polar coordinate system. Polar_angle is ALWAYS two-theta, even if the detector does not happen to live in the scattering plane. The azimuthal_angle defines by how much the plane in which the 0 point of the drawing and the blue point live is tilted towards the scattering plane.

`polar_angle`: rotation x' about the X axis
`azimuthal_angle`: rotation z' about the Z axis
`rotation_angle`: rotation y' about the Y axis

Note
The NeXus definition of `+z` is opposite to that in the International Tables for Crystallography, volume G, and consequently, `+x` is also reversed.

2.3.1 Simple (Spherical Polar) Coordinate System

In this system, the instrument is considered as a set of components through which the incident beam passes. The variable **distance** is assigned to each component and represents the effective beam flight path length between this component and the sample. A sign convention is used where negative numbers represent components pre-sample and positive numbers components post-sample.

For angular information, the quantities *polar_angle* and *azimuthal_angle*, as shown in Figure 2.1, are used and these quantities correspond exactly to the usual spherical polar coordinate definitions i.e. the `polar_angle` is the *zenith angle* and measured with respect to a z axis and the `azimuthal_angle` to the x axis in the xy plane. The direction of these local axes may be different for each component: z is the incident beam direction for the **previous** component and we then follow *McStas* for x and y i.e. the x axis is perpendicular to the beam in the horizontal plane pointing left as seen from the source, and the y axis points upwards (see diagram below). The z axis thus represents the direction of the beam if it was not redirected by the previous component, and so the polar_angle and azimuthal_angle for a component indicate how much the beam was bent/scattered by the previous component. The `rotation_angle` is a rotation about the y axis.

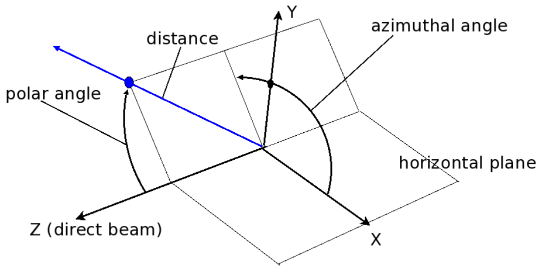

Figure 2.1: NeXus Simple (Spherical Polar) Coordinate System

If we consider an `NXdetector` element placed directly after an NXsample, the z axis would be in the direction of the beam incident on NXsample. The polar_angle for the `NXdetector` would be the angle between the scattered beam and this z axis and so correspond to the *Bragg angle* or *two theta* even for out-of-plane scattering. The azimuthal_angle would be the angle between the positive x-axis and the scattered beam projected onto the xy-plane - scattering to the left as seen from the source would have azimuthal_angle=0 and scattering to the right azimuthal_angle=pi. The distance would correspond to what is often called the *secondary flight path length* or *L2*.

2.3.2 `NXgeometry`-based system

The `NXgeometry`-based coordinate system, shown in Figure 2.2, is based more fully on the *McStas coordinate system.* The instrument is given a global, absolute coordinate system where the z axis points in the direction of the incident beam, the x axis is perpendicular to the beam in the horizontal plane pointing left as seen from the source, and the *y* axis points upwards. Each beamline component also has a local coordinate system, which is defined by the NXgeometry object. The local *z* direction for a component is taken as the incident beam direction, with *x* and *y* defined as before (*i.e.* the *x* axis is perpendicular to the beam in the horizontal plane pointing left as seen from the source, and the *y* axis points upwards). Information about these coordinate systems and the placement of components is described by the NXgeometry class via its `NXtranslation` and `NXorientation` members.

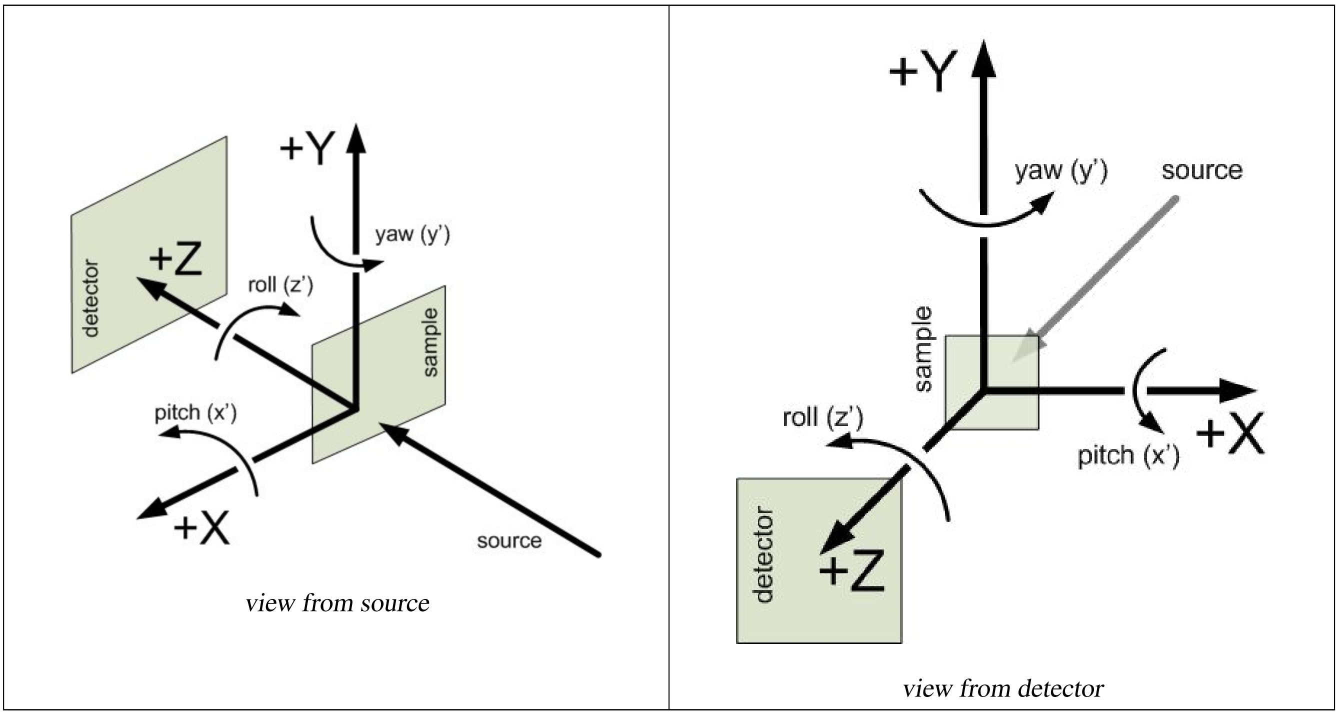

Figure 2.2: `NXgeometry`-based coordinate system

The sign of the rotations as shown in Figure 2.2, is technically correct although instrument scientists might debate whether it is sensible. (For example, an increasing Bragg angle for a vertical reflection where the beam is bouncing upwards seems as if it should be positive but the proper sense of this *x'*, or pitch, rotation must be negative if one wishes to obey the right-hand rule.) A review of what is written in the International Tables for Crystallography volume G might be in order to resolve this.

2.3.2.1 Coordinate Transformations

When computing a transformation, `NXtranslation` is applied before `NXorientation`. All NeXus axes are right handed and orthogonal. Orientation information is stored as direction cosines. The direction cosines will be between the local coordinate directions and the reference directions (to origin or relative NXgeometry). From the `value` field in `NXorientation`, ' Calling the local unit vectors *(x',y',z')* and the reference unit vectors *(x,y,z)* the six numbers will be *[x' dot x, x' dot y, x' dot z, y' dot x, y' dot y, y' dot z]* where "dot" is the scalar dot product (cosine of the angle between the unit vectors). The unit vectors in both the local and reference coordinates are right-handed and orthonormal. ' With this restriction we only need to store 6 rather than 9 direction cosines, as the *z'* axis can be obtained by the vector cross product of *x'* and *y'*.

2.3.2.2 The Coordinate Origin

The origin of coordinates is arbitrary, but all components in the file must either agree on its absolutelocation or use relative positioning. To allow for this generality, an origin member can be defined in `NXentry`; its use will be detailed shortly.

We choose as our absolute the origin the scattering center, which is where a perfectly aligned sample would be. Note that the centre of the sample itself may not always be at this point if the sample is being scanned across the beam. With an origin at the *scattering centre* the spherical polar coordinate specifications of the detector positions conveniently relates to scattering angles and lengths for direct geometry instruments.

Individual components of the instrument (e.g. jaws) will have their own set of local axes (x,y,z) which will be fixed to their body in a way defined by their shape. These local axes will probably not coincidewith the global instrument axes and so a set of rotation angles will also need to be stored. For this an NXgeometry class is defined, along with `NXtranslation` and `NXorientation`; the hope is to provide a method sufficiently general for relating the location of any object with respect to another object. The mechanism also allows for specifying one position relative to another component: a NeXus file link is made in one instance of an NXgeometry object to another NXgeometry object and a program can then traverse the chain of links to calculate an absolute position.

NeXus does not need to define absolutely where to place the *origin*. All components can instead be declared with a relative position that ultimately follows a chain back to one object of class NXgeometry; this will be named *origin1*, be and a *field* of `NXentry`. The real space location of this origin is chosen for convenience and should be mentioned in the description attached to `origin1`. If the origin is taken at the sample, then `sample.geometry.distance` will always be *(0,0,0)* relative to `origin1`; if the origin is taken elsewhere this will not be so, but everything will still work. It may be convenient to define extra origins (similar to *arms* in *McStas*) at other parts of the instrument. For example, defining one at the centre of a circular array of detectors would allow their positions to be conveniently specified in spherical polar coordinates. Another possibility would be to define the sample relative to *origin1* and the detectors to *origin2*; the detectors could then be rotated by a rotation of *origin2* without modifying `NXdetector`.

As well as specifying the component location, it is also necessary to specify the beam direction. Unless otherwise given in an `NXbeam` member of the component, the incident beam is assumed to be travelling along *(0,0,+z)* in the coordinate system of the object (or origin) our position was defined relative to. Thus, for a component with absolute positioning the beam will always be in the incident beam direction unless specified by an `NXbeam` member.

2.3.2.3 Size and Shape (NXshape)

Many instrument components define variables to specify their size. For example, *radius* might be used to specify a circular object while *height* and *width* might be used to specify a rectangular object. Rather than specify all these different names, an alternative scheme is proposed based on the *shape* of the object and the local coordinate axes this shape defines. All objects would just need to specify a shape (*cuboid*, *cylinder* etc.) and a size array. Specifying `size[3]` would give the dimensions of the object along its local *(+-x,+-y,+-z)* axes; specifying `size[6]` would give the extent along *(+x,+y,+z,-x,-y,-z)* and allow for e.g. asymmetric jaws where the reference point may not be the centre of the rectangle.

For example take `shape="cylinder"`: the `NXtranslation` variable of position would define the location of the reference point for the origin of the local axes: *z* in the direction of the cylinder axis, *x* and *y* in plane. With no rotation, the object would be oriented with its local axes pointing in the direction of axes of the object it was defined relative to, but this can be altered with the `NXorientation` variable within position. If a `size[3]` array variable was specified, the reference point must be the centre of the cylinder and the dimension are `size[0]=size[1]=radius`, `size[2]=length/2`. If `size[6]` was specified then the reference point would be elsewhere in the object, with its distance from the cylinder edges along the various axes given by elements of the `size[6]` array.

2.4 NeXus units

NeXus units are written as a string (`NX_CHAR`) and describe the engineering units. The string should be appropriate for the value. Values for the NeXus units must be specified in a format compatible with Unidata UDunits.[2] The UDunits specification also includes instructions for derived units. utility (in particular, see the now-deprecated `udunits.txt`).[3] At present, the contents of NeXus `units` attributes are not validated.

[2] Unidata UDunits: `http://www.unidata.ucar.edu/software/udunits/`
[3] (deprecated): `http://www.unidata.ucar.edu/software/udunits/udunits-1/udunits.txt`

2.5 NeXus dates and times

NeXus dates and times should be stored using the ISO 8601[4] format e.g. `1996-07-31 21:15:22+0600`. The standard also allows for time intervals in fractional seconds with *1 or more digits of precision*. This will avoid confusion, e.g. between U.S. and European conventions, and is appropriate for machine sorting.

2.6 NeXus array dimensions

Note

Need a better writeup than this! Also need to confirm if this is correct!

Here are a couple of examples to get this section started:

Example 2.4 Example of array dimensions.

```
<field name="data">
  <dimensions size="3"/>
  <!-- other definitions might appear -->
</field><field name="time-of-flight">
  <dimensions size="1">
    <dim index="1" ref="data" refindex="3" incr="1"/>
  </dimensions>
  <!-- other definitions might appear -->
</field>
```

In some programming language, this would make data[i,j,k] and time-of-flight[k+1].

2.7 NeXus Data Types

glean this information from `NeXus.xsd`

2.8 Rules for Storing Data in NeXus Files

What are the rules?

[4] ISO 8601: http://www.w3.org/TR/NOTE-datetime

Chapter 3

The NeXus Application-Program Interface

3.1 Description of the NeXus Application-Program Interface

The NeXus API was written to shield (and hide) the complexity of the HDF API from scientific programmers and users of the NeXus Data Standard.

For a more detailed description of the internal workings of NAPI that is maintained (mostly) concurrent with code revisions, see `NeXusIntern.pdf`[1] in the NeXus code repository. Likely this is only interesting for programmers who wish to hack the NAPI.

In Section 1.5, there is the mention of "a more complete version" of "a typical NeXus program written in C." Where is it?

[1] `NeXusIntern.pdf`: `http://svn.nexusformat.org/code/trunk/doc/api/NeXusIntern.pdf`

Chapter 4

The NeXus Definition Language

Need to edit this into shape. Work though `nxdl.xsd` for ideas and content.

The NeXus Design page lists the group classes from which a NeXus file is constructed. They provide the glossary of items that could, in principle, be stored in a standard-conforming NeXus file (other items may be inserted into the file if the author wishes, but they won't be part of the standard). If you are going to include a particular piece of metadata, consult the class definitions to find out what to call it. However, to assist those writing data analysis software, it is useful to provide more than a glossary; it is important to define the required contents of NeXus files that contain data from particular classes of neutron, x-ray, or muon instrument.

As part of the NeXus standard, we have identified a number of generic instruments that describe an appreciable number of existing instruments around the world. Although not identical in every detail, they share enough common characteristics, and more importantly, they require sufficiently similar modes of data analysis, to make a standard description useful. They are in the process of being defined for the NeXus standard. The definitions will be in XML using the NeXus Definition Language (NXDL) format.

4.1 comments from Trac ticket #65

1. **Application definitions** will override the standard definition of (the base class) `NXentry` and provide fields and groups that *must* be present in a NeXus data file. Other fields or groups from `NXentry` (or other base classes) are optional, as usual.

2. Some mechanism needs to exist within the `NXroot` base class to identify the use of an application definition to use instead of the standard `NXentry`.

   One suggestion is to add an optional `NXDL` attribute to the `NXentry` group entry where the value of the attribute is the name of the application definition. This preserves Mark Koennecke's suggestion: ‘ [The reference to the defining NXDL] needs to be at a standard place across all files and it needs to live below `NXentry` as a NeXus file may contain different entries adhering to different definitions. ’ For example:

Example 4.1 suggestion to identify the use of an application definition

```
<NXentry name="SCAN_0001" NXDL="NXmonopd">
    <!-- the NXmonopd data comes here -->
</NXentry>]]>
```

   In a followup, Freddie Akeroyd said: ‘ With regard to [this point], isn't this already covered by the **definition** field in the `NXentry`? ’

   ‘ I think there may be an option to use `xsi:type` at the `NXentry` level to asist with schema validation of the NXDL translated files ... I've been updating the `definition` option of `nxconvert` to add such an attribute so we would have [something like the next example] to validate against a schema. If `NXmonopd` is inherited from `NXentry`, then `xsi:type` will tell the schema validation process to use `NXmonopd` rather than `NXentry` when it is run. '

Example 4.2 use of xsi:type to reference a defining NXmonopd

```
<NXentry name="SCAN_0001" xsi:type="NXmonopd">
    <definition url="...">NXmonopd</definition>
</NXentry>
```

Pete added this comment:

- keywords NXDL validation Schema NXentry added
  - NXentry has a definition field that specifies the NXDL to be used.
  - For validation of any NeXus data file, consider this transformation scenario:
    * (Advance step) Prepare XML Schema files from NXDL
    * (Advance step) Collect all XML Schema files into a master XML Schema
    * For HDF data, extract most content (except for data) to NeXus XML data file format
    * Transform XML data file to XML data validation file format (this is not done yet)
    * Validate XML data validation file against XML Schema
  - Optional fields or groups are used in an application definition to declare nomenclature

3. An application definition shall contain the **minimum set of information necessary to do common processing like data analysis.** This also means that there is no space in the application definition for optional fields.
   But of course, in a real NeXus data file the user is always permitted to add additional fields from the base classes.

4. The original NeXus view was that a base class was just a dictionary of allowable terms (hence everything optional) and a definition said which base class terms were mandatory (with those not specified being treated as optional). Thus a definition did not need to add optional items as this was covered by the base class.

5. Perhaps this was never written down, but may have been an implication that a definition could not mark a field as mandatory if it was not already listed in some base class as optional.

6. A base class has a wide range of terms, some of which would make more sense in a particular instrument/application definition than others. Thus, there is some merit from the documentation point of view in having some optional items in a definition as *things you might want to consider adding* or even the specification of the name if that optional term is to be used.

4.2 short comment to MAHID group on 2010-01-26

"NeXus Application Definition" is meant to describe NXDL specifications for scientific techniques and instrument definitions.

Class definitions in NeXus prior to 2008 had been in the form of base classes and instrument definitions. All of these were in the same category. As the development of NeXus had been led mostly by scientists from neutron sources, this represented their typical situations.

Both those new to NeXus and also those familiar saw the previous emphasis on instrument definitions as a deficiency that limited flexibility and possibly usage. The point was made that NeXus should attempt to better describe reduced data and also data for analysis since synchrotron instruments are rarely adhering to a fixed definition.

The design of NeXus is moving towards an object-oriented approach where the base classes will be the objects and the "application definitions" will use the objects to specify the required components as fits some application. Here, "application" is very loosely defined to include:

- specification of a scientific instrument (example: TOF-USANS at SNS)
- specification of what is expected for a scientific technique (example: small-angle scattering data for comm
- specification of generic data acquisition stream (example: TOFRAW - raw time-of-flight data from a pulse
- specification of input or output of a specific software program

The term "the sky is the limit" seems to apply. The point of the "NeXus Application Definition" is that all of these start with "NX" and all have been approved by the NIAC.

Those NXDL specifications not yet approved by the NIAC fall into the category of "NeXus contributed definitions" for which NeXus has a place in the repository. At present, this place is empty. Think of this category as place to put an NXDL (a candidate for a base class or application definition) for the NIAC to consider approving.

4.3 Description of the NeXus Definition Language

The intent of the NeXus Definition Language (NXDL) is to provide an easier, and rules-based method for defining a NeXus data file that is specific to either an instrument (where NeXus has been for years) or an area of scientific technique or analysis.

NeXus wiki page: NXDL[1]

NXDL is the new NeXus definition language - it will replace the Meta-DTD format[2] as the way for specifying the content of NeXus data files. NXDL is based on XML schema technology and follows on from provisional work on NeXus schema NeXus schema[3] in Summer 2008.

At NIAC 2008,[4] it was agreed that writing definitions directly as schema would probably be too verbose and instead a new language (NXDL) was devised that would then be translated into XML schema for eventual validation. The files written in the NXDL language are XML files in their own right and have an associated XML schema - this means a schema-aware XML editor can be used and accurate definition creation is much easier than it was with the old Meta-DTD.

For the latest NXDL definitions see the svn repository.[5]

An NXDL description will be a true (not pseudo) XML file which structure can be validated by a schema. See below for a draft example from the working repository. Since the NXDL specification is not complete, expect that some aspects of this example might change. NXDL is not intended to change the location of information stored in existing NeXus files, only to change (and simplify) the way the file would be arranged for a specific instance such as instrument or technique.

There are several different elements used in the NXDL. These are:

Table 4.1: Tabular representation of NXuser[definition]

Name and Attributes (Type)	Description (and Occurrences)
`definition` (XML root element)	Root element of an NXDL specification. This is the basic definition of a new type. Components must derive from an existing type and optionally provide documentation.
`@name` ()	required
`@type` (`nx:definitionTypeAttr`)	required
`@extends` ()	optional
`@restricts` ()	optional
`@svnid` ()	optional
`group`	A group element refers to the definition of an existing NX object: either a base class (such as `NXdata` or `NXgeometry`), application definition, contributed definition, or a locally-defined component.
`field`	A field declares a new element in the component being defined.
`doc`	Documentation for the field or group. The item can include DocBook formatting but it is necessary to make the DocBook namespace default for any formatting elements. For example: <para xmlns="http://docbook.org/ns/docbook">
`link`	A link to another item.
`attribute`	attribute of `field`

Note

An application definition specifies the *minimum set* of data information (both data and metadata) for data analysis software.

[1] http://www.nexusformat.org/NXDL
[2] http://www.nexusformat.org/Metaformat
[3] http://www.nexusformat.org/Schema
[4] http://www.nexusformat.org/images/b/b2/NIAC2008_minutes.pdf
[5] https://svn.nexusformat.org/definitions/trunk

4.3.1 NXbending_magnet.nxdl.xml: example NXDL specification

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
# NeXus - Neutron and X-ray Common Data Format
#
# Copyright (C) 2008 NeXus International Advisory Committee (NIAC)
#
# This library is free software; you can redistribute it and/or
# modify it under the terms of the GNU Lesser General Public
# License as published by the Free Software Foundation; either
# version 2 of the License, or (at your option) any later version.
#
# This library is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
# Lesser General Public License for more details.
#
# You should have received a copy of the GNU Lesser General Public
# License along with this library; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
#
# For further information, see http://www.nexusformat.org

########### SVN repository information ###################
# $Date: 2010-01-29 18:02:12 -0600 (Fri, 29 Jan 2010) $
# $Author: Pete Jemian $
# $Revision: 480 $
# $HeadURL: https://svn.nexusformat.org/definitions/trunk/base_classes/NXbending_magnet. ↩
    nxdl.xml $
# $Id: NXbending_magnet.nxdl.xml 480 2010-01-30 00:02:12Z Pete Jemian $
########### SVN repository information ###################
-->
<definition xmlns="http://definition.nexusformat.org/nxdl/3.1" category="base"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://definition.nexusformat.org/nxdl/3.1 ../nxdl.xsd"
    name="NXbending_magnet"
    version="1.0"
    svnid="$Id: NXbending_magnet.nxdl.xml 480 2010-01-30 00:02:12Z Pete Jemian $"
  type="group" extends="NXobject">

  <doc>description for a bending magnet</doc>
  <field name="critical_energy" type="NX_FLOAT" units="NX_ENERGY"/>
  <field name="bending_radius" type="NX_FLOAT" units="NX_LENGTH"/>
  <group name="spectrum" type="NXdata"><doc>bending magnet spectrum</doc></group>
  <group type="NXgeometry"><doc>"Engineering" position of bending magnet</doc></group>
</definition>
```

4.3.2 nxdl.xsd source code

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
########### SVN repository information ###################
# $LastChangedDate: 2010-02-06 22:15:57 -0600 (Sat, 06 Feb 2010) $
# $LastChangedBy: Pete Jemian $
# $LastChangedRevision: 495 $
# $HeadURL: https://svn.nexusformat.org/definitions/trunk/nxdl.xsd $
########### SVN repository information ###################
-->

```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://definition.nexusformat.org/nxdl/3.1"
  xmlns:nx="http://definition.nexusformat.org/nxdl/3.1"
  version="$Id: nxdl.xsd 495 2010-02-07 04:15:57Z Pete Jemian $"
  elementFormDefault="qualified">

  <xs:annotation>
    <xs:documentation>
      # NeXus - Neutron and X-ray Common Data Format
      #
      # Copyright (C) 2008-2010 NeXus International Advisory Committee (NIAC)
      #
      # This library is free software; you can redistribute it and/or
      # modify it under the terms of the GNU Lesser General Public
      # License as published by the Free Software Foundation; either
      # version 2 of the License, or (at your option) any later version.
      #
      # This library is distributed in the hope that it will be useful,
      # but WITHOUT ANY WARRANTY; without even the implied warranty of
      # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
      # Lesser General Public License for more details.
      #
      # You should have received a copy of the GNU Lesser General Public
      # License along with this library; if not, write to the Free Software
      # Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
      #
      # For further information, see http://www.nexusformat.org
    </xs:documentation>
  </xs:annotation>

  <!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->

  <xs:include schemaLocation="nxdlTypes.xsd">
    <xs:annotation>
      <xs:documentation>
        Definitions of the basic data types and unit types
        allowed in NXDL instance files.
      </xs:documentation>
    </xs:annotation>
  </xs:include>

  <xs:simpleType name="validItemName">
    <xs:annotation>
         <xs:documentation>
          Used for allowed names of elements and attributes.
          Need to be restricted to valid program variable names.
          Note:  This means no "-" or "." characters can be allowed and
          you cannot start with a number.
          HDF4 had a 64 character limit on names
          (possibly including NULL) and NeXus enforces this
          via the NX_MAXNAMELEN variable.
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:pattern value="[A-Za-z_][\w_]*" />
      <xs:maxLength value="63" />   <!-- enforce via NX_MAXNAMELEN -->
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="validNXClassName">
    <xs:annotation>
         <xs:documentation>
```

```
      Used for allowed names of NX class types (e.g. NXdetector)
      not the instance (e.g. bank1) which is covered by validItemName.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="nx:validItemName">
      <xs:pattern value="NX.+"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="validTargetName">
  <xs:annotation>
    <xs:documentation>
      This is a valid link target - currently it must be an absolute path
      made up of valid names with / character separating, but we may
      want to consider allowing ".." at some point.
      Must also consider use of name attribute in resolving link targets.
      /NXentry/NXinstrument/NXcrystal:analyzer/ef
      /NXentry/NXinstrument/NXcrystal:monochromator/ei
      /NX_other
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:token">
    <xs:annotation>
      <xs:documentation>
        The HDF5 documentation
        (http://www.hdfgroup.org/HDF5/doc/UG/UG_frame09Groups.html)
        says "Note that relative path names in HDF5 do not employ the ../ notation,
        the UNIX notation indicating a parent directory, to indicate a parent group."
        Thus, if we only consider the case of
        <code xmlns="http://docbook.org/ns/docbook">
          class[:name]
        (/[a-zA-Z_][\w_]*(:[a-zA-Z_][\w_]*)?)+
        </code>
        Note that HDF5 also permits relative path names, such as:
        <code xmlns="http://docbook.org/ns/docbook">
          GroupA/GroupB/Dataset1
        </code>
        but this is not permitted in the pattern below and not supported in NAPI.
      </xs:documentation>
    </xs:annotation>
    <xs:pattern value="(/[a-zA-Z_][\w_]*(:[a-zA-Z_][\w_]*)?)+" />
  </xs:restriction>
</xs:simpleType>


<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->

<!-- define the document root element -->
<xs:element name="definition" type="nx:definitionType" />

<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->

<xs:complexType name="definitionType">
  <xs:group ref="nx:groupGroup" minOccurs="0" maxOccurs="unbounded" />
  <xs:attribute name="name" use="required" type="nx:validItemName" />
  <xs:attribute name="version" use="required" />
  <xs:attribute name="type" use="required" type="nx:definitionTypeAttr" />
  <xs:attribute name="extends" use="optional" />
  <xs:attribute name="restricts" use="optional" />
  <xs:attribute name="svnid" use="optional"/>
  <xs:attribute name="category" use="required">
    <xs:simpleType>
```

```
        <xs:restriction base="xs:string">
          <xs:enumeration value="base"/>
          <xs:enumeration value="application"/>
          <xs:enumeration value="contributed"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>

  <xs:complexType name="attributeType">
    <xs:annotation>
      <xs:documentation>
        A new element may expect or require some attributes.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="doc" type="nx:docType" minOccurs="0" />
      <xs:element name="enumeration" type="nx:enumerationType"
        minOccurs="0">
        <xs:annotation>
          <xs:documentation>An enumeration specifies the values to be used.</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" use="required" type="nx:validItemName" />
    <xs:attribute name="type" type="nx:primitiveType" default="NX_CHAR"/>
  </xs:complexType>

  <xs:simpleType name="definitionTypeAttr">
    <xs:restriction base="xs:string">
      <xs:enumeration value="group" />
      <xs:enumeration value="definition" />
    </xs:restriction>
  </xs:simpleType>

  <xs:group name="groupGroup">
    <xs:sequence>
      <xs:element name="doc" type="nx:docType" minOccurs="0"
        maxOccurs="1" />
      <xs:element name="attribute" type="nx:attributeType"
        minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="group" type="nx:groupType" minOccurs="0"
        maxOccurs="unbounded" />
      <xs:element name="field" type="nx:fieldType" minOccurs="0"
        maxOccurs="unbounded" />
      <xs:element name="link" type="nx:linkType" minOccurs="0"
        maxOccurs="unbounded" />
    </xs:sequence>
  </xs:group>

  <!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->

  <xs:complexType name="basicComponent">
    <xs:sequence>
      <xs:element name="doc" type="nx:docType" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="name" use="required" type="nx:validItemName" />
  </xs:complexType>

  <!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->

```

```
  <xs:simpleType name="basicType">
    <xs:annotation>
      <xs:documentation>
        This is the basic definition of a new type.
        Components must derive from an existing type
        and optionally provide documentation.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"></xs:restriction>
  </xs:simpleType>

  <!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->

  <xs:complexType name="groupType">
    <xs:annotation>
      <xs:documentation>
        A group element refers to the definition of
        an existing NX object or a locally-defined component.
      </xs:documentation>
    </xs:annotation>
    <xs:group ref="nx:groupGroup" minOccurs="0" maxOccurs="unbounded"/>
    <xs:attribute name="type" use="required" type="nx:validNXClassName"/>
    <xs:attribute name="minOccurs" use="optional" default="0" type="xs:nonNegativeInteger"/>
    <xs:attribute name="maxOccurs" use="optional"  default="unbounded"/>
    <xs:attribute name="name" use="optional" type="nx:validItemName"/>
  </xs:complexType>

  <!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->

  <xs:complexType name="dimsType">
    <!-- =================================================================== -->
    <!-- ====================== BEGIN MAJOR COMMENT ========================== -->
    <!-- =================================================================== -->
    <!--
Peter Peterson writes:
      Here is a model:
      <field name="data">
          <dimensions size="3"/>
          ...
      </field><field name="time-of-flight">
        <dimensions size="1">
          <dim index="1" ref="data" refindex="3" incr="1"/>
        </dimensions>
        ...
      </field>

      In old terms this would make data[i,j,k] and time-of-flight[k+1].
      We need to get to the stage where people do not
      need to read a manual to understand an existing NXDL.

+++++++++++++++++++++++++++++++++++++
then Freddie Akeroyd writes:
I think we have three cases to consider here:
(1) specifying an explicit value for a dimension
(2) specifying a dimension to be the same size as that of an array
already defined elsewhere (optionally +- a value)
(3) specifying a dimension to be the same as a data value defined
elsewhere (e.g. num_scan_points, optionally +- a value)

For (1) how about

```

```
<dim index="1" value="3" />

For (2) how about

<field name="data">
<dimensions size="3">
<dim index="1" label="num_time_channels"/>
< !- - dimensions with no constraints or labels do not need to be mentioned - -  >
</dimensions>
</field>
<field name="time_of_flight">
<dimensions size="1">
<dim index="1" ref="num_time_channels" incr="1"/>
</dimensions>
...
</field>

For (3) how about

<field name="num_scan_points" type="NX_INT"
label="nscanpt" />
<field name="data">
<dimensions size="3">
<dim index="1" label="num_time_channels"/>
<dim index="3" ref="nscanpt"/>
</dimensions>
</field>

Notes on dim tag:

use 1 for first element in a sequence, +- to indicate sequence direction
(or we could use an attribute instead)

<dim index="1">   ! refers to first index (fastest varying)
<dim index="-1">  ! refers to last index (slowest varying)

<dim label="i">   ! we should constrain labels to be unique within the
file and thus could be referred to
from any location. This would mean that "i"
could not be used everywhere as a label,
but forcing the use of better names like
"num_time_channels" makes things clearer. In the
first instance by making label an xs:ID and
ref an xs:IDREF we can enforce "something" in
label="something" to always be unique and "some_ref" in ref="some_ref"
to always point to a
valid "label" (We may later want to use xs:key
/ xs:keyref to do this if we needed a second
set of unique labels for another purpose)

+++++++++++++++++++++++++++++++++++++
then Mark Koennecke writes:
I think the dimensions encoding is tightly integrated with the algorithm we
use to validate them later on. I thought about something like:

<dimensions size="3">
<dim level="0" value="25"/>
<dim level="1" value="np"/>
<dim level="2" path="/entry/blablabla/somefield" dim="0"/>
</dimensions>
The first case is trivial: validate against a number

```

```
The general idea is that the dimensions validator would initialise np
the first place it comes up against it and
checks against the value ever after. We must allow expressions like np+1
here.

The third syntax is to explicitly address a specific dimension of
another variable.

    -->

    <!-- ===================================================================== -->
    <!-- ======================= END MAJOR COMMENT =========================== -->
    <!-- ===================================================================== -->
    <xs:annotation>
      <xs:documentation>dimensions of a data element in a NeXus file</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="dim" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="index" type="nx:NX_CHAR">
            <xs:annotation>
              <xs:documentation>Number indicating which axis (subscript) is
                being described, ranging from 1 up to "size" (rank of the
                data structure).  For example, given an array A[i,j,k],
                index="1" would refer to the "i" axis (subscript).</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="value" type="nx:NX_CHAR">
            <xs:annotation>
              <xs:documentation>Length (number of values) of this axis.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="ref" type="nx:NX_CHAR">
            <xs:annotation>
              <xs:documentation>The dimension specification is the same as
                that in the "ref" field, specified either by a relative path,
                such as "polar_angle" or "../Qvec" or absolute path, such as
                "/entry/path/to/follow/to/ref/field".
              </xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="refindex" type="nx:NX_CHAR">
            <xs:annotation>
              <xs:documentation>The dimension specification is the same as
                the "refindex" axis within the "ref" field.
                Requires "ref" attribute to be present.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="incr" type="nx:NX_CHAR">
            <xs:annotation>
              <xs:documentation>The dimension specification is related to
                the "refindex" axis within the "ref" field by an
                offset of "incr."  Requires "ref" and "refindex"
                attributes to be present.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="size" type="nx:NX_CHAR">
      <xs:annotation>
        <xs:documentation>Rank (number of dimensions) of the data structure.
```

```
          For example: a[5] has size="1" while b[8,5,6,4] has size="4".
          See http://en.wikipedia.org/wiki/Rank_(computer_programming)
          for more details.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>

  <xs:complexType name="linkType">
    <xs:complexContent>
      <xs:annotation>
        <xs:documentation>A link to another item.</xs:documentation>
      </xs:annotation>
      <xs:extension base="nx:basicComponent">
        <xs:attribute name="target" use="required" type="nx:validTargetName">
          <xs:annotation>
            <xs:documentation>path to element we link to</xs:documentation>
          </xs:annotation>
        </xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="fieldType">
    <xs:complexContent>
      <xs:annotation>
        <xs:documentation>A field declares a new element
        in the component being defined.</xs:documentation>
      </xs:annotation>
      <xs:extension base="nx:basicComponent">
        <xs:sequence>
          <xs:element name="dimensions" type="nx:dimsType"
            minOccurs="0" maxOccurs="1">
            <xs:annotation>
              <xs:documentation>dimensions of a data element
              in a NeXus file</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="attribute" type="nx:attributeType"
            minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
              <xs:documentation>attributes of field</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="enumeration" type="nx:enumerationType"
            minOccurs="0">
            <xs:annotation>
              <xs:documentation>A field can specify which
              values are to be used</xs:documentation>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="units" type="nx:anyUnitsAttr">
          <xs:annotation>
            <xs:documentation>
              String describing the engineering units.
              The string should be appropriate for the value
              and should conform to the NeXus rules for units.
              Can conformance be validated or ensured?
            </xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="signal" type="nx:NX_POSINT">
```

```
            <xs:annotation>
              <xs:documentation>
                Presence of the signal attribute means this field is an ordinate.

                Integer marking this field as plottable data (ordinates).
                The value indicates the priority of selection or interest.
                Some facilities only use signal="1"
                while others use signal="2" to indicate
                plottable data of secondary interest.
                Higher numbers are possible but not common
                and interpretation is not standard.

                A field with a signal attribute should not have an axis attribute.
              </xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="axes" type="nx:NX_CHAR">
            <xs:annotation>
              <xs:documentation>
                Presence of the axes attribute means this field is an ordinate.

                This attribute contains a white-space separated list
                of paths to the names of independent axes when plotting this field.
              </xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="axis" type="nx:NX_POSINT">
            <xs:annotation>
              <xs:documentation>
                NOTE: Use of this attribute is discouraged.  It is for legacy support.
                You should use the axes attribute instead.

                Presence of the axis attribute means this field is an abcissa.

                Integer marking this
                field as an axis that is part of the data set.
                The data set is a field with the attribute
                signal="1" in the same group.
                The value can range from 1 up to the number of
                independent axes (abcissae) in the data set.

                A value of axis="1" indicates that this field
                contains the data for the first independent axis.
                For example, the X axis in an XY data set.

                A value of axis="2" indicates that this field
                contains the data for the second independent axis.
                For example, the Y axis in a 2-D data set.

                A value of axis="3" indicates that this field
                contains the data for the third independent axis.
                For example, the Z axis in a 3-D data set.

                A field with an axis attribute should not have a signal attribute.
              </xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="primary" type="nx:NX_POSINT">
            <xs:annotation>
              <xs:documentation>
                Integer indicating the priority of selection
                of this field for plotting (or visualization) as an axis.
```

```
              Presence of the primary attribute means this field is an abcissa.
            </xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="type" type="nx:primitiveType" default="NX_CHAR"/>
        <xs:attribute name="minOccurs" use="optional" default="0">
          <xs:simpleType>
            <xs:restriction base="xs:nonNegativeInteger">
              <xs:enumeration value="0"/>
              <xs:enumeration value="1"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="maxOccurs" use="optional"  default="1">
          <xs:simpleType>
            <xs:restriction base="xs:nonNegativeInteger">
              <xs:enumeration value="1"/>
              <!--
                The "unbounded" is not valid for nonNegativeInteger
                But that is what we really mean.
              -->
              <xs:enumeration value="999999999999999999999999"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->

  <xs:complexType name="docType" mixed="true">
    <xs:annotation>
      <xs:documentation>
        This is the definition of optional documentation.
        Be sure to use the DocBook namespace for any DocBook elements:
        xmlns="http://docbook.org/ns/docbook"
      </xs:documentation>
    </xs:annotation>
    <!--<xs:restriction base="nx:basicType">-->
    <!--</xs:restriction>-->
    <xs:sequence>
      <xs:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="enumerationType">
    <xs:sequence>
      <xs:element name="item" type="nx:enumItemType" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="enumItemType">
    <xs:sequence>
      <xs:element name="doc" type="nx:docType" minOccurs="0" maxOccurs="1">
        <xs:annotation>
          <xs:documentation>
            Items can be documented
            but enumeration documentation may not
            printed in the manual.
            Still OK to use DocBook elements.
```

```
            Be sure to use the DocBook namespace for any DocBook elements:
            xmlns="http://docbook.org/ns/docbook"
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="value" use="required"/>
  </xs:complexType>

</xs:schema>
```

4.3.3 Data Types allowed in NXDL specification

Data Types for use in NXDL specifications describe the expected type of data for a NeXus field. These terms are very broad. More specific terms are used in actual NeXus data files that describe size and array dimensions. In addition to the types in the following table, the `NAPI` type is defined when one wishes to permit a field with any of these data types.

Table 4.2: Data Types allowed in NXDL specifications

term	description
`NX_CHAR`	any string representation
`NX_FLOAT`	any representation of a floating point number
`NX_INT`	any representation of an integer number
`NX_UINT`	any representation of an unsigned integer number (includes zero)
`NX_POSINT`	any representation of a positive integer number (greater than zero)
`NX_NUMBER`	any valid NeXus number representation
`NX_DATE_TIME`	alias for the ISO8601 date/time stamp
`NX_BOOLEAN`	true/false value
`NX_BINARY`	any representation of binary data - if text, line terminator is [CR][LF]

4.3.4 Unit Categories allowed in NXDL specifications

Unit categories in NXDL specifications describe the expected type of units for a NeXus field. They should describe valid units consistent with the section on NeXus units (Section 2.4). The values for unit categories are restricted (by an enumeration) to the following table.

Table 4.3: Unit Types allowed in NXDL specifications

term	description
`NX_ANGLE`	example: degrees or radians or arcminutes or
`NX_ANY`	usage: things like logs that aren't picky on units
`NX_AREA`	example: m2 or barns
`NX_CROSS_SECTION`	example: barns
`NX_CURRENT`	example: A
`NX_DIMENSIONLESS`	for fields where the units cancel out, example: "" or mm/mm (NOTE: not the same as NX_UNITLESS)
`NX_ENERGY`	example: J or keV
`NX_FLUX`	example: s-1 cm-2
`NX_FREQUENCY`	example: Hz
`NX_LENGTH`	example: m
`NX_MASS`	example: g

Table 4.3: (continued)

term	description
NX_MASS_DENSITY	example: g cm-3
NX_MOLECULAR_WEIGHT	example: g mol-1
NX_PER_AREA	example: cm-2
NX_PER_LENGTH	example: cm-1
NX_PERIOD	(alias to NX_TIME) period of pulsed source, example: microseconds
NX_POWER	example: W
NX_PRESSURE	example: Pa
NX_PULSES	(alias to NX_NUMBER) clock pulses
NX_SCATTERING_LENGTH_DENSITY	example: cm-2
NX_SOLID_ANGLE	example: sr | steradian
NX_TEMPERATURE	example: K
NX_TIME	example: s
NX_TIME_OF_FLIGHT	(alias to NX_TIME) example: s
NX_VOLTAGE	example: V
NX_VOLUME	example: m3
NX_UNITLESS	for fields that don't have a unit (e.g. hkl) so that they don't inherit the wrong units (NOTE: not the same as NX_DIMENSIONLESS)
NX_WAVELENGTH	example: Angstrom
NX_WAVENUMBER	units for Q, example: Angstrom-1 or nm-1

Chapter 5

Verification and validation of files

The intent of verification and validation of files is to ensure, in an unbiased way, that a given file conforms to the relevant specifications. NeXus uses various automated tools to validate files. These tools include conversion of content from HDF to XML and transformation (via XSLT) from XML format to another such as NXDL, XSD, and Schematron. This chapter will first provide an overview of the process, then define the terms used in validation, then describe how multiple base classes or application definitions might apply to a given NeXus data file, and then describe the various validation techniques in more detail. Validation does not check that the data content of the file is sensible; this requires scientific interpretation based on the technique.

Validation is useful to anyone who manipulates or modifies the contents of NeXus files. This includes scientists/users, instrument staff, software developers, and those who might mine the files for metadata. First, the scientist or user of the data must be certain that the information in a file can be located reliably. The instrument staff or software developer must be confident the information they have written to the file has been located and formatted properly. At some time, the content of the NeXus file may contribute to a larger body of work such as a metadata catalog for a scientific instrument, a laboratory, or even an entire user facility.

5.1 Overview

NeXus files adhere to a set of rules and can be tested against these rules for compliance. The rules are implemented using standard tools and can themselves be tested to verify compliance with the standards for such definitions. Validation includes the testing of both NeXus data files and the NXDL specifications that describe the rules.

Note
Needs drawing showing how the validation process works.

NeXus data files

NeXus data files (also known as NeXus data file instances) are validated to ensure the various parts of the data file are arranged according to the governing NXDL specifications used in that file instance.

NeXus Definition Language (NXDL) specification files

NXDL files are validated to ensure they adhere to the rules for writing NeXus base classes and application definitions.

5.2 Definitions of these terms

This is a paragraph.

5.3 NeXus data files may use multiple base classes or application definitions

This is a paragraph.

5.4 Validation techniques

Describe the tools used to validate files from the user's perspective and then from the software developer's perspective.

5.4.1 Overview

This is a paragraph.

5.4.2 Validation of NeXus data files

Each NeXus data file can be validated against the NXDL rules. (The full suite of NXDL specifications is converted into Schematron rules by an XSLT transformation and then combined into a single file. It is not allowed to have a NeXus base class and also an application definition with the same name since one will override the other in the master Schematron file) The validation is done using Schematron and the `NXvalidate` program. Schematron was selected, rather than XML Schema (XSD), to permit established rules for NeXus files, especially the rule allowing the nodes within `NXentry` to appear in any order.

Note

It is very important to describe what is and is not being validated.

First, the NeXus data file instance (either HDF or XML) is converted

5.4.3 Validation of NeXus Definition Language (NXDL) specification files

Each NXDL file must be validated against the rules that define how NXDL files are to be arranged. The rules are specified in the form of XML Schema (XSD).

5.4.4 Schematron

This is a paragraph.

5.4.5 Transformation of NXDL files to Schematron

This is a paragraph.

Appendix A

NeXus classes

A.1 Overview of NeXus classes

Each of the NeXus classes is described in two basic ways. First, a short list of descriptive information is provided as a header, then a table summarizing the fields and groups that comprise the NeXus class is given.

category The category of NXDL, either:

`base` (base class)
`application` (application definition)
`contributed` (contributed definition)

NXDL source Name of the NeXus class and a URL to the source listing in the NeXus subversion repository.

version A string that documents this particular version of this NXDL.

SVN Id Subversion repository checkout identification, stripped of the surrounding dollar signs. (The *Id* is blank on files copied direct from the repository that are not checked out by a subversion client.)

NeXus Definition Language The NeXus Definition Language (NXDL) (described in a separate chapter of this manual) is used to describe the components in the NeXus Base Classes, as well as application and contributed definitions. The intent of NXDL is to provide a rules-based method for defining a NeXus data file that is specific to either an instrument (where NeXus has been for years) or an area of scientific technique or analysis. NXDL replaces the meta-DTD method used previously to define the NeXus base classes.

extends class NeXus class extended by this class. Most NeXus base classes only extend the base class definition (NXDL).

other classes included List (including URLs) of other classes used to define this class.

documentation Description of the NeXus class. No markup or formatting is allowed.

The table has columns to describe the basic information about each field or group in the class. An example of the varieties of specifications are given in the following table using items found in various NeXus base classes.

Table A.1: Example Tabular representation of a NeXus class

Name	Type	Units	Description (and Occurences)
`program_name`	NX_CHAR		Name of program used to generate this file
`@version`	NX_CHAR		Program version number Occurences: 1 : *default*
`@configuration`	NX_CHAR		configuration of the program
`thumbnail`	NXnote		A small image that is representative of the entry. An example of this is a 640x480 JPEG image automatically produced by a low resolution plot of the NXdata.

Table A.1: (continued)

Name	Type	Units	Description (and Occurences)
@mime_type	NX_CHAR		expected: **mime_type="image/*"**
	NXgeometry		describe the geometry of this class
distance	NX_FLOAT	NX_LENGTH	Distance from sample
mode	"Single Bunch" | "Multi Bunch"		source operating mode
target_material	Ta | W | depleted_U | enriched_U | Hg | Pb | C		Pulsed source target material

In the above example, the fields might appear in a NeXus XML data file as

Example A.1 Example fragment of a NeXus XML data file

```
<program_name version="1.0a" configuration="standard">
    MaxSAS
</program_name>
<NXnote name="thumbnail" mime_type="image/*">
    <!-- contents of an NXnote would appear here -->
</NXnote>
<distance units="mm">125.6</distance>
<mode> Single Bunch </mode>
<target_material>depleted_U</target_material>
```

The columns in the table are described as follows:

Name (and attributes) Name of the data field. Since `name` needs to be restricted to valid program variable names, no "-" characters can be allowed. Name must satisfy both HDF and XML naming rules.

```
NameStartChar ::=  _ | a..z | A..Z
NameChar      ::=  NameStartChar | 0..9
Name          ::=  NameStartChar (NameChar)*

Or, as a regular expression:  [_a-zA-Z][_a-zA-Z0-9]*
```

Attributes, identified with a leading "at" symbol (`@`) and belong with the preceding field or group, are additional metadata used to define this field or group.

In the example above, the `program_name` element has two attributes: `version` (required) and `configuration` (optional) while the `thumbnail` element has one attribute: `mime_type` (optional).

Type Type of data to be represented by this variable. The type is one of those specified in the NeXus Definition Language (see Chapter 4).

In the case where the variable can take only one value from a known list, the list of known values is presented, such as in the `target_material` field above: `Ta | W | depleted_U | enriched_U | Hg | Pb | C` . Selections with included whitespace are surrounded by quotes. See the example above for usage.

Units Data units, given as character strings, must conform to the NeXus units standard. See the "NeXus units" section for details.

Description (and Occurences) A simple text description of the data field. No markup or formatting is allowed.

The absence of *Occurences* in the item description signifies that both `minOccurs` and `maxOccurs` have the default values. If the number of occurences of an item are specified in the NXDL (through `@minOccurs` and `@maxOccurs` attributes), they will be reported in the Description column similar to the example shown above.

Default values for occurences are shown in the following table. The `NXDL element type` is either a group (such as a NeXus base class), a field (that specifies the name and type of a variable), or an attribute of a field or group. The number of times an item can appear ranges between `minOccurs` and `maxOccurs`. A default `minOccurs` of zero means the item is optional. For attributes, `maxOccurs` cannot be greater than 1.

Table A.2: Default values for occurences

NXDL element type	minOccurs	maxOccurs
group	0	unbounded
field	0	unbounded
attribute	0	1

A.2 NeXus Base Classes

A description of each NeXus Base Class is given.

A.2.1 NXaperture

category `base` (base class)

NXDL source: NXaperture (http://svn.nexusformat.org/definitions/trunk/base_classes/NXaperture.nxdl.xml)

version 1.0

SVN Id Id: NXaperture.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXgeometry

documentation Template of a beamline aperture.

Table A.3: Tabular representation of NXaperture[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
	NXgeometry		location and shape of aperture
`material`	NX_CHAR		Absorbing material of the aperture
`description`	NX_CHAR		Description of aperture

A.2.2 NXattenuator

category `base` (base class)

NXDL source: NXattenuator (http://svn.nexusformat.org/definitions/trunk/base_classes/NXattenuator.nxdl.xml)

version 1.0

SVN Id Id: NXattenuator.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: no included classes

documentation Template of a beamline attenuator.

Table A.4: Tabular representation of NXattenuator[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
distance	NX_FLOAT	NX_LENGTH	Distance from sample
type	NX_CHAR		Type of attenuator, e.g. polythene
thickness	NX_FLOAT	NX_LENGTH	Thickness of attenuator along beam direction
scattering_cross_section	NX_FLOAT	NX_CROSS_SECTION	Scattering cross section (coherent+incoherent)
absorption_cross_section	NX_FLOAT	NX_CROSS_SECTION	Absorption cross section
attenuator_transmission	NX_FLOAT	NX_DIMENSIONLESS	The nominal amount of the beam that gets through (transmitted intensity)/(incident intensity)

A.2.3 NXbeam

category base (base class)

NXDL source: NXbeam (http://svn.nexusformat.org/definitions/trunk/base_classes/NXbeam.nxdl.xml)

version 1.0

SVN Id Id: NXbeam.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata

documentation Template of the state of the neutron or X-ray beam at any location. It will be referenced by beamline component groups within the NXinstrument group or by the NXsample group. Note that variables such as the incident energy could be scalar values or arrays. This group is especially valuable in storing the results of instrument simulations in which it is useful to specify the beam profile, time distribution etc. at each beamline component. Otherwise, its most likely use is in the NXsample group in which it defines the results of the neutron scattering by the sample, e.g., energy transfer, polarizations.

Table A.5: Tabular representation of NXbeam[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
distance	NX_FLOAT	NX_LENGTH	Distance from sample
incident_energy	NX_FLOAT	NX_ENERGY	Energy on entering beamline component Dimensions: size="1" • dim: index="1" value="i" value=""

Table A.5: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
final_energy	NX_FLOAT	NX_ENERGY	Energy on leaving beamline component Dimensions: size="1" • dim: index="1" value="i" value=""
energy_transfer	NX_FLOAT	NX_ENERGY	Energy change caused by beamline component Dimensions: size="1" • dim: index="1" value="i" value=""
incident_wavel-ength	NX_FLOAT	NX_WAVELENGTH	Wavelength on entering beamline component Dimensions: size="1" • dim: index="1" value="i" value=""
incident_wavel-ength_spread	NX_FLOAT	NX_WAVELENGTH	Wavelength spread FWHM on entering component Dimensions: size="1" • dim: index="1" value="i" value=""
incident_beam_-divergence	NX_FLOAT	NX_ANGLE	Divergence of beam entering this component Dimensions: size="2" • dim: index="1" value="2" value="" • dim: index="2" value="j" value=""
final_waveleng-th	NX_FLOAT	NX_WAVELENGTH	Wavelength on leaving beamline component Dimensions: size="1" • dim: index="1" value="i" value=""
incident_polar-ization	NX_FLOAT	NX_ANY	Polarization vector on entering beamline component Dimensions: size="2" • dim: index="1" value="2" value="" • dim: index="2" value="j" value=""
final_polariza-tion	NX_FLOAT	NX_ANY	Polarization vector on leaving beamline component Dimensions: size="2" • dim: index="1" value="2" value="" • dim: index="2" value="j" value=""
final_waveleng-th_spread	NX_FLOAT	NX_WAVELENGTH	Wavelength spread FWHM of beam leaving this component Dimensions: size="1" • dim: index="1" value="i" value=""
final_beam_div-ergence	NX_FLOAT	NX_ANGLE	Divergence FWHM of beam leaving this component Dimensions: size="2" • dim: index="1" value="2" value="" • dim: index="2" value="j" value=""
flux	NX_FLOAT	NX_FLUX	flux incident on beam plane area Dimensions: size="1" • dim: index="1" value="i" value=""

Table A.5: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
	NXdata		Distribution of beam with respect to relevant variable e.g. wavelength. This is mainly useful for simulations which need to store plottable information at each beamline component.

A.2.4 NXbeam_stop

category `base` (base class)

NXDL source: NXbeam_stop (http://svn.nexusformat.org/definitions/trunk/base_classes/NXbeam_stop.nxdl.xml)

version 1.0

SVN Id Id: NXbeam_stop.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXgeometry

documentation A class for a beamstop. Beamstops and their positions are important for SANS and SAXS experiments.

Table A.6: Tabular representation of NXbeam_stop[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
	NXgeometry		engineering shape, orientation and position of the beam stop.
`description`	circular | rectangular		description of beamstop
`size`	NX_FLOAT	NX_LENGTH	size of beamstop
`x`	NX_FLOAT	NX_LENGTH	x position of the beamstop in relation to the detector
`y`	NX_FLOAT	NX_LENGTH	y position of the beamstop in relation to the detector
`distance_to_detector`	NX_FLOAT	NX_LENGTH	distance of the beamstop to the detector
`status`	in | out		

A.2.5 NXbending_magnet

category `base` (base class)

NXDL source: NXbending_magnet (http://svn.nexusformat.org/definitions/trunk/base_classes/NXbending_magnet.nxdl.xml)

version 1.0

SVN Id Id: NXbending_magnet.nxdl.xml 480 2010-01-30 00:02:12Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXgeometry

documentation description for a bending magnet

Table A.7: Tabular representation of NXbending_magnet[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
critical_energy	NX_FLOAT	NX_ENERGY	
bending_radius	NX_FLOAT	NX_LENGTH	
spectrum	NXdata		bending magnet spectrum
	NXgeometry		"Engineering" position of bending magnet

A.2.6 NXcharacterization

category base (base class)

NXDL source: NXcharacterization (http://svn.nexusformat.org/definitions/trunk/base_classes/NXcharacterization.nxdl.xml)

version 1.0

SVN Id Id: NXcharacterization.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: no included classes

documentation Template of the top-level NeXus group which contains all the data and associated information that comprise a single measurement. It is mandatory that there is at least one group of this type in the NeXus file.

Table A.8: Tabular representation of NXcharacterization[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
definition	NX_CHAR		
@version	NX_CHAR		
@URL	NX_CHAR		

A.2.7 NXcollimator

category base (base class)

NXDL source: NXcollimator (http://svn.nexusformat.org/definitions/trunk/base_classes/NXcollimator.nxdl.xml)

version 1.0

SVN Id Id: NXcollimator.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXgeometry, NXlog

documentation Template of a beamline collimator.

Table A.9: Tabular representation of NXcollimator[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
	NXgeometry		position, shape and size
type	Soller | radial | oscillating | honeycomb		
soller_angle	NX_FLOAT	NX_ANGLE	Angular divergence of Soller collimator
divergence_x	NX_FLOAT	NX_ANGLE	divergence of collimator in local x direction
divergence_y	NX_FLOAT	NX_ANGLE	divergence of collimator in local y direction
frequency	NX_FLOAT	NX_FREQUENCY	Frequency of oscillating collimator
frequency_log	NXlog		Log of frequency
blade_thickness	NX_FLOAT	NX_LENGTH	blade thickness
blade_spacing	NX_FLOAT	NX_LENGTH	blade spacing
absorbing_material	NX_CHAR		name of absorbing material
transmitting_material	NX_CHAR		name of transmitting material

A.2.8 NXcrystal

category base (base class)

NXDL source: NXcrystal (http://svn.nexusformat.org/definitions/trunk/base_classes/NXcrystal.nxdl.xml)

version 1.0

SVN Id Id: NXcrystal.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXgeometry, NXlog

documentation Template of a crystal monochromator or analyzer. Permits double bent monochromator comprised of multiple segments with anisotropic Gaussian mosaic. If curvatures are set to zero or are absent, array is considered to be flat. Scattering vector is perpendicular to surface. Crystal is oriented parallel to beam incident on crystal before rotation, and lies in vertical plane.

Table A.10: Tabular representation of NXcrystal[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
	NXgeometry		Position of crystal
type	PG | Ge | Si | Cu | Fe3Si | CoFe | Cu2MnAl | Multilayer | Diamond		Material of monochromating substance. Use the "reflection" field to indicate the (hkl) orientation. Use the "d_spacing" field to record the lattice plane spacing.
cut_angle	NX_FLOAT	NX_ANGLE	Cut angle of reflecting Bragg plane and plane of crystal surface

Table A.10: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
unit_cell	NX_FLOAT	NX_LENGTH	Unit cell parameters (lengths and angles) Dimensions: size="2" • dim: index="1" value="n_comp" value="" • dim: index="2" value="6" value=""
unit_cell_volu-me	NX_FLOAT	NX_VOLUME	Volume of the unit cell
orientation_ma-trix	NX_FLOAT		Orientation matrix of single crystal sample using Busing-Levy convention Dimensions: size="2" • dim: value="3" value="" • dim: value="3" value=""
wavelength	NX_FLOAT	NX_WAVELENGTH	Optimum diffracted wavelength Dimensions: apparent size="1" • dim: value="i" value=""
d_spacing	NX_FLOAT	NX_LENGTH	spacing between crystal planes of the reflection
scattering_vec-tor	NX_FLOAT	NX_WAVENUMBER	Scattering vector, Q, of nominal reflection
reflection	NX_INT	NX_UNITLESS	Miller indices (hkl) values of nominal reflection Dimensions: apparent size="1" • dim: value="3" value=""
segment_width	NX_FLOAT	NX_LENGTH	Horizontal width of individual segment
segment_height	NX_FLOAT	NX_LENGTH	Vertical height of individual segment
segment_thickn-ess	NX_FLOAT	NX_LENGTH	Thickness of individual segment
segment_gap	NX_FLOAT	NX_LENGTH	Typical gap between adjacent segments
segment_columns	NX_FLOAT	NX_LENGTH	number of segment columns in horizontal direction
segment_rows	NX_FLOAT	NX_LENGTH	number of segment rows in vertical direction
mosaic_horizon-tal	NX_FLOAT	NX_ANGLE	horizontal mosaic Full Width Half Maximum
mosaic_vertical	NX_FLOAT	NX_ANGLE	vertical mosaic Full Width Half Maximum
curvature_hori-zontal	NX_FLOAT	NX_ANGLE	Horizontal curvature of focusing crystal
curvature_vert-ical	NX_FLOAT	NX_ANGLE	Vertical curvature of focusing crystal
polar_angle	NX_FLOAT	NX_ANGLE	Polar (scattering) angle at which crystal assembly is positioned. Note: some instrument geometries call this term 2theta. Dimensions: apparent size="1" • dim: ref="wavelength" value=""
azimuthal_angle	NX_FLOAT	NX_ANGLE	Azimuthal angle at which crystal assembly is positioned Dimensions: apparent size="1" • dim: ref="wavelength" value=""
bragg_angle	NX_FLOAT	NX_ANGLE	Bragg angle of nominal reflection Dimensions: apparent size="1" • dim: ref="wavelength" value=""
temperature	NX_FLOAT	NX_TEMPERATURE	average/nominal crystal temperature

Table A.10: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
temperature_log	NXlog		log file of crystal temperature
reflectivity	NXdata		crystal reflectivity versus wavelength
transmission	NXdata		crystal transmission versus wavelength

A.2.9 NXdata

category base (base class)

NXDL source: NXdata (http://svn.nexusformat.org/definitions/trunk/base_classes/NXdata.nxdl.xml)

version 1.0

SVN Id Id: NXdata.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: no included classes

documentation Template of plottable data and their dimension scales. It is mandatory that there is at least one group of this type in each NXentry group. Note that "variable" and "data" can be defined with different names. The "signal" and "axes" attribute of the "data" item define which items are plottable data and which are dimension scales.

Table A.11: Tabular representation of NXdata[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
variable	NX_NUMBER		Dimension scale defining an axis of the data Dimensions: size="1" • dim: index="1" value="i" value=""
@long_name	NX_CHAR		Axis label
@distribution	NX_BOOLEAN		0|false: single value, 1:true: multiple values
@first_good	NX_INT		Index of first good value
@last_good	NX_INT		Index of last good value
@axis	NX_POSINT		Index (positive integer) identifying this specific set of numbers
variable_errors	NX_NUMBER		Errors associated with axis "variable" Dimensions: size="1" • dim: index="1" value="i" value=""
data	NX_NUMBER		Data values Dimensions: size="1" • dim: index="1" value="i" value=""
@signal	NX_INT		plottable (independent) axis, indicate index number
@axes	NX_CHAR		names of dependent axes
@long_name	NX_CHAR		data label
errors	NX_NUMBER		Standard deviations of data values - the data array is identified by the attribute signal="1". This array must have the same dimensions as the data Dimensions: size="1" • dim: index="1" value="i" value=""

Table A.11: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
scaling_factor	NX_FLOAT		The elements in data are usually float values really. For efficiency reasons these are usually stored as integers after scaling with a scale factor. This value is the scale factor. It is required to get the actual physical value, when necessary.
offset	NX_FLOAT		An optional offset to apply to the values in data.
x	NX_FLOAT	NX_ANY	This is an array holding the values to use for the x-axis of data. The units must be appropriate for the measurement. Dimensions: size="1" • dim: index="1" value="nx" value=""
y	NX_FLOAT	NX_ANY	This is an array holding the values to use for the y-axis of data. The units must be appropriate for the measurement. Dimensions: size="1" • dim: index="1" value="ny" value=""
z	NX_FLOAT	NX_ANY	This is an array holding the values to use for the z-axis of data. The units must be appropriate for the measurement. Dimensions: size="1" • dim: index="1" value="nz" value=""

A.2.10 NXdetector

category base (base class)

NXDL source: NXdetector (http://svn.nexusformat.org/definitions/trunk/base_classes/NXdetector.nxdl.xml)

version 1.0

SVN Id Id: NXdetector.nxdl.xml 473 2010-01-26 22:56:14Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXcharacterization, NXdata, NXgeometry, NXnote

documentation Template of a detector, detector bank, or multidetector. The indices require explanation:

np the number of points in a scan. This dimension is only present in scanning measurements.

tof the number of points in a scan. This dimension is only present in scanning measurements.

i the number of pixels in the slowest varying direction. This is only missing in the point detector.

j the number of pixels in the fastest varying direction. This exists only in "area" detectors.

Table A.12: Tabular representation of NXdetector[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
time_of_flight	NX_FLOAT	NX_TIME_OF_FLIGHT	Total time of flight Dimensions: size="1" • dim: index="1" value="tof+1" value=""
@axis	NX_INT		
@primary	NX_INT		
@long_name	NX_CHAR		Axis label
@link	NX_CHAR		absolute path to location in NXdetector
raw_time_of_fl-ight	NX_INT	NX_PULSES	In DAQ clock pulses Dimensions: size="1" • dim: index="1" value="tof+1" value=""
@frequency	NX_FLOAT		Clock frequency in Hz
detector_number	NX_INT		Identifier for detector Dimensions: size="2" • dim: index="1" value="i" value="" • dim: index="2" value="j" value=""
data	NX_NUMBER	NX_ANY	Data values Dimensions: size="4" • dim: index="1" value="np" value="" • dim: index="2" value="i" value="" • dim: index="3" value="j" value="" • dim: index="4" value="tof" value=""
@signal	NX_INT		
@axes	NX_CHAR		[number of scan points,x_offset?,y_offset?,time_of_flight?]
@long_name	NX_CHAR		Title of measurement
@check_sum	NX_INT		Integral of data as check of data integrity
@link	NX_CHAR		absolute path to location in NXdetector
data_error	NX_NUMBER	NX_ANY	Data values Dimensions: size="4" • dim: index="1" value="np" value="" • dim: index="2" value="i" value="" • dim: index="3" value="j" value="" • dim: index="4" value="tof" value=""
@units	NX_CHAR		
@link	NX_CHAR		absolute path to location in NXdetector
x_pixel_offset	NX_FLOAT	NX_LENGTH	offset from the detector center in x-direction Dimensions: size="1" • dim: index="1" value="i" value=""
@axis	NX_INT		
@primary	NX_INT		
@long_name	NX_CHAR		Axis label
@link	NX_CHAR		absolute path to location in NXdetector
y_pixel_offset	NX_FLOAT	NX_LENGTH	offset from the detector center in the y-direction Dimensions: size="1" • dim: index="1" value="j" value=""
@axis	NX_INT		

Table A.12: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
@primary	NX_INT		
@long_name	NX_CHAR		Axis label
distance	NX_FLOAT	NX_LENGTH	need some documentation here Dimensions: size="3" • dim: index="1" value="np" value="" • dim: index="2" value="i" value="" • dim: index="3" value="j" value=""
polar_angle	NX_FLOAT	NX_ANGLE	need some documentation here Dimensions: size="3" • dim: index="1" value="np" value="" • dim: index="2" value="i" value="" • dim: index="3" value="j" value=""
azimuthal_angle	NX_FLOAT	NX_ANGLE	need some documentation here Dimensions: size="3" • dim: index="1" value="np" value="" • dim: index="2" value="i" value="" • dim: index="3" value="j" value=""
description	NX_CHAR		name/manufacturer/model/etc. information
local_name	NX_CHAR		Local name for the detector
	NXgeometry		Position and orientation of detector element
solid_angle	NX_FLOAT	NX_SOLID_ANGLE	Solid angle subtended by the detector at the sample Dimensions: size="2" • dim: index="1" value="i" value="" • dim: index="2" value="j" value=""
x_pixel_size	NX_FLOAT	NX_LENGTH	Size of each detector pixel. If it is scalar all pixels are the same size Dimensions: size="2" • dim: index="1" value="i" value="" • dim: index="2" value="j" value=""
y_pixel_size	NX_FLOAT	NX_LENGTH	Size of each detector pixel. If it is scalar all pixels are the same size Dimensions: size="2" • dim: index="1" value="i" value="" • dim: index="2" value="j" value=""
dead_time	NX_FLOAT	NX_TIME	Detector dead time Dimensions: size="3" • dim: index="1" value="np" value="" • dim: index="2" value="i" value="" • dim: index="3" value="j" value=""
gas_pressure	NX_FLOAT	NX_PRESSURE	Detector gas pressure Dimensions: size="2" • dim: index="1" value="i" value="" • dim: index="2" value="j" value=""

Table A.12: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
`detection_gas_path`	NX_FLOAT	NX_LENGTH	maximum drift space dimension
`crate`	NX_INT		Crate number of detector Dimensions: size="2" • dim: index="1" value="i" value="" • dim: index="2" value="j" value=""
`@local_name`	NX_CHAR		Equivalent local term
`slot`	NX_INT		Slot number of detector Dimensions: size="2" • dim: index="1" value="i" value="" • dim: index="2" value="j" value=""
`@local_name`	NX_CHAR		Equivalent local term
`input`	NX_INT		Input number of detector Dimensions: size="2" • dim: index="1" value="i" value="" • dim: index="2" value="j" value=""
`@local_name`	NX_CHAR		Equivalent local term
`type`	NX_CHAR		Description of type such as He3 gas cylinder, He3 PSD, scintillator, fission chamber, proportion counter, ion chamber, ccd, pixel, image plate, cmos, ...
`efficiency`	NXdata		Efficiency of detector with respect to e.g. wavelength Look for special case table **efficiency[NXdata](within NXdetector[definition])** below.
`calibration_date`	NX_DATE_TIME		date of last calibration (geometry and/or efficiency) measurements
`calibration_method`	NXnote		summary of conversion of array data to pixels (e.g. polynomial approximations) and location of details of the calibrations
`layout`	point | linear | area		How the detector is represented
`count_time`	NX_NUMBER	NX_TIME	Elapsed actual counting time Dimensions: size="1" • dim: index="1" value="np" value=""
`data_file`	NXnote		
	NXcharacterization		
`sequence_number`	NX_CHAR		In order to properly sort the order of the images taken in (for example) a tomography experiment, a sequence number is stored with each image. Dimensions: size="1" • dim: index="1" value="nBrightFrames" value=""

A.2.10.1 Special case table: efficiency[NXdata](within NXdetector[definition])

Table A.13: Tabular representation of efficiency[NXdata](within NXdetector[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
efficiency	NX_FLOAT	NX_DIMENSIONLESS	efficiency of the detector Dimensions: size="3" • dim: index="1" value="i" value="" • dim: index="2" value="j" value="" • dim: index="3" value="k" value=""
wavelength	NX_FLOAT	NX_WAVELENGTH	need some documentation here Dimensions: size="3" • dim: index="1" value="i" value="" • dim: index="2" value="j" value="" • dim: index="3" value="k" value=""

A.2.11 NXdetector_group

category base (base class)

NXDL source: NXdetector_group (http://svn.nexusformat.org/definitions/trunk/base_classes/NXdetector_group.nxdl.xml)

version 1.0

SVN Id Id: NXdetector_group.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: no included classes

documentation This class is used to allow a logical grouping of detector elements (e.g. which tube, bank or group of banks) to be recorded in the file. As well as allowing you to e.g just select the "left" or "east" detectors, it may also be useful for determining which elements belong to the same PSD tube and hence have e.g. the same dead time. For example, if we had "bank1" composed of "tube1", "tube2" and "tube3" then group_names would be the string "bank1, bank1/tube1, bank1/tube2,bank1/tube3" group_index would be {1,2,3,4} group_parent would be {-1,1,1,1} The mapping array is interpreted as group 1 is a top level group containing groups 2, 3 and 4 A group_index array in NXdetector give the base group for a detector element.

Table A.14: Tabular representation of NXdetector_group[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
group_names	NX_CHAR		Comma separated list of name
group_index	NX_INT		Unique ID for group. A group_index array in NXdetector give the base group for a detector element. Dimensions: apparent size="1" • dim: value="i" value=""
group_parent	NX_INT		Index of group parent in the hierarchy, -1 means no parent (i.e. a top level) group Dimensions: apparent size="1" • dim: ref="group_index" value=""

Table A.14: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
group_type	NX_INT		Code number for group type e.g. bank=1, tube=2 etc. Dimensions: apparent size="1" • dim: ref="group_index" value=""

A.2.12 NXdisk_chopper

category base (base class)

NXDL source: NXdisk_chopper (http://svn.nexusformat.org/definitions/trunk/base_classes/NXdisk_chopper.nxdl.xml)

version 1.0

SVN Id Id: NXdisk_chopper.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXgeometry

documentation

Table A.15: Tabular representation of NXdisk_chopper[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
type	"Chopper type single" | contra_rotating_pair | synchro_pair		
rotation_speed	NX_FLOAT	NX_FREQUENCY	chopper rotation speed
slits	NX_INT		Number of slits
slit_angle	NX_FLOAT	NX_ANGLE	angular opening
pair_separation	NX_FLOAT	NX_LENGTH	disc spacing in direction of beam
radius	NX_FLOAT	NX_LENGTH	radius to centre of slit
slit_height	NX_FLOAT	NX_LENGTH	total slit height
phase	NX_FLOAT	NX_ANGLE	chopper phase angle
ratio	NX_INT		pulse reduction factor of this chopper in relation to other choppers/fastest pulse in the instrument
distance	NX_FLOAT	NX_LENGTH	Effective distance to the origin
wavelength_range	NX_FLOAT	NX_WAVELENGTH	low and high values of wavelength range transmitted Dimensions: apparent size="1" • dim: value="2" value=""
	NXgeometry		

A.2.13 NXentry

category base (base class)

NXDL source: NXentry (http://svn.nexusformat.org/definitions/trunk/base_classes/NXentry.nxdl.xml)

version 1.0

SVN Id Id: NXentry.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXcharacterization, NXdata, NXinstrument, NXmonitor, NXnote, NXprocess, NXsample, NXuser

documentation Template of the top-level NeXus group which contains all the data and associated information that comprise a single measurement. It is mandatory that there is at least one group of this type in the NeXus file.

Table A.16: Tabular representation of NXentry[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
title	NX_CHAR		Extended title for entry
experiment_identifier	NX_CHAR		Unique identifier for the experiment, defined by the facility, possibly linked to the proposals
experiment_description	NX_CHAR		Brief summary of the experiment, including key objectives.
experiment_documentation	NXnote		Description of the full experiment (document in pdf, latex, . . .)
collection_identifier	NX_CHAR		User or Data Acquisition defined group of NeXus files or NXentry
collection_description	NX_CHAR		Brief summary of the collection, including grouping criteria.
entry_identifier	NX_CHAR		unique identifier for the measurement, defined by the facility.
definition @version @URL	NX_CHAR NX_CHAR NX_CHAR		Official NeXus DTD or NXDL schema to which this file conforms DTD or NXDL version number URL of DTD or NXDL file
definition_local @version @URL	NX_CHAR NX_CHAR NX_CHAR		Local DTD or NXDL schema extended from the file specified in the "definition" tag. This contains the locally defined additional fields in the file. DTD or NXDL version number URL of DTD or NXDL file
start_time	NX_DATE_TIME		Starting time of measurement
end_time	NX_DATE_TIME		Ending time of measurement
duration	NX_INT	NX_TIME	Duration of measurement
collection_time	NX_FLOAT	NX_TIME	Time transpired actually collecting data i.e. taking out time when collection was suspended due to e.g. temperature out of range
run_cycle	NX_CHAR		
program_name @version @configuration	NX_CHAR NX_CHAR NX_CHAR		Name of program used to generate this file Program version number configuration of the program
revision @comment	NX_CHAR NX_CHAR		Revision id of the file due to re-calibration, reprocessing, new analysis, new instrument definition format, ...
notes	NXnote		Notes describing entry
thumbnail @mime_type	NXnote NX_CHAR		A small image that is representative of the entry. An example of this is a 640x480 jpeg image automatically produced by a low resolution plot of the NXdata. The value should be an "image/*"
	NXcharacterization		
	NXuser		
	NXsample		
	NXinstrument		

Table A.16: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
	NXmonitor		
	NXdata		
	NXprocess		

A.2.14 NXenvironment

category `base` (base class)

NXDL source: NXenvironment (http://svn.nexusformat.org/definitions/trunk/base_classes/NXenvironment.nxdl.xml)

version 1.0

SVN Id Id: NXenvironment.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXgeometry, NXnote, NXsensor

documentation This class describes an external condition applied to the sample

Table A.17: Tabular representation of NXenvironment[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
`name`	NX_CHAR		Apparatus identification code/model number; e.g. OC100 011
`short_name`	NX_CHAR		Alternative short name, perhaps for dashboard display like a present Seblock name
`type`	NX_CHAR		Type of apparatus. This could be the SE codes in scheduling database; e.g. OC/100
`description`	NX_CHAR		Description of the apparatus; e.g. 100mm bore orange cryostat with Roots pump
`program`	NX_CHAR		Program controlling the apparatus; e.g. LabView VI name
`position`	NXgeometry		The position and orientation of the apparatus
	NXnote		Additional information, LabView logs, digital photographs, etc
	NXsensor		

A.2.15 NXevent_data

category `base` (base class)

NXDL source: NXevent_data (http://svn.nexusformat.org/definitions/trunk/base_classes/NXevent_data.nxdl.xml)

version 1.0

SVN Id Id: NXevent_data.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: no included classes

documentation Time-of-flight events

Table A.18: Tabular representation of NXevent_data[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
`time_of_flight`	NX_INT	NX_TIME_OF_FLIGHT	A list of time of flight for each event as it comes in. This list is for all pulses with information to attach to a particular pulse located in events_per_pulse. Dimensions: size="1" • dim: index="1" value="i" value=""
`pixel_number`	NX_INT	NX_DIMENSIONLESS	There will be extra information in the NXdetector to convert pixel_number to detector_number. This list is for all pulses with information to attach to a particular pulse located in events_per_pulse. Dimensions: size="1" • dim: index="1" value="i" value=""
`pulse_time`	NX_INT	NX_TIME	The time that each pulse started with respect to the offset Dimensions: size="1" • dim: index="1" value="j" value=""
`@offset`	NX_INT		ISO8601
`events_per_pul-se`	NX_INT	NX_DIMENSIONLESS	This connects the index "i" to the index "j". The jth element is the number of events in "i" that occurred during the jth pulse. Dimensions: size="1" • dim: index="1" value="j" value=""
`pulse_height`	NX_FLOAT	NX_DIMENSIONLESS	If voltages from the ends of the detector are read out this is where they go. This list is for all events with information to attach to a particular pulse height. The information to attach to a particular pulse is located in events_per_pulse. Dimensions: size="2" • dim: index="1" value="i" value="" • dim: index="2" value="k" value=""

A.2.16 NXfermi_chopper

category `base` (base class)

NXDL source: NXfermi_chopper (http://svn.nexusformat.org/definitions/trunk/base_classes/NXfermi_chopper.nxdl.xml)

version 1.0

SVN Id Id: NXfermi_chopper.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXgeometry

documentation Description of a Fermi chopper, possibly with curved slits.

Table A.19: Tabular representation of NXfermi_chopper[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
type	NX_CHAR		Fermi chopper type
rotation_speed	NX_FLOAT	NX_FREQUENCY	chopper rotation speed
radius	NX_FLOAT	NX_LENGTH	radius of chopper
slit	NX_FLOAT	NX_LENGTH	width of an individual slit
r_slit	NX_FLOAT	NX_LENGTH	radius of curvature of slits
number	NX_INT	NX_UNITLESS	number of slits
height	NX_FLOAT	NX_LENGTH	input beam height
width	NX_FLOAT	NX_LENGTH	input beam width
wavelength	NX_FLOAT	NX_WAVELENGTH	Wavelength transmitted by chopper
	NXgeometry		geometry of the fermi chopper
absorbing_mate-rial	NX_CHAR		absorbing material
transmitting_m-aterial	NX_CHAR		transmitting material

A.2.17 NXfilter

category base (base class)

NXDL source: NXfilter (http://svn.nexusformat.org/definitions/trunk/base_classes/NXfilter.nxdl.xml)

version 1.0

SVN Id Id: NXfilter.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXgeometry, NXlog, NXsensor

documentation Template for specifying the state of band pass filters

Table A.20: Tabular representation of NXfilter[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
	NXgeometry		Geometry of the filter
description	Beryllium | "Pyrolytic Graphite" | Graphite | Sapphire | Silicon | Supermirror		
status	in | out		
transmission	NXdata		Wavelength transmission profile of filter
temperature	NX_FLOAT	NX_TEMPERATURE	average/nominal filter temperature
temperature_log	NXlog		Linked temperature_log for the filter

Table A.20: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
sensor_type	NXsensor		Sensor(s)used to monitor the filter temperature
unit_cell_a	NX_FLOAT	NX_LENGTH	Unit cell lattice parameter: length of side a
unit_cell_b	NX_FLOAT	NX_LENGTH	Unit cell lattice parameter: length of side b
unit_cell_c	NX_FLOAT	NX_LENGTH	Unit cell lattice parameter: length of side c
unit_cell_alpha	NX_FLOAT	NX_ANGLE	Unit cell lattice parameter: angle alpha
unit_cell_beta	NX_FLOAT	NX_ANGLE	Unit cell lattice parameter: angle beta
unit_cell_gamma	NX_FLOAT	NX_ANGLE	Unit cell lattice parameter: angle gamma
unit_cell_volume	NX_FLOAT	NX_VOLUME	Unit cell Dimensions: size="1" • dim: index="1" value="n_comp" value=""
orientation_matrix	NX_FLOAT		Orientation matrix of single crystal filter Dimensions: size="3" • dim: index="1" value="n_comp" value="" • dim: index="2" value="3" value="" • dim: index="3" value="3" value=""
m_value	NX_FLOAT	NX_DIMENSIONLESS	m value of supermirror filter
substrate_material	NX_CHAR		substrate material of supermirror filter
substrate_thickness	NX_FLOAT	NX_LENGTH	substrate thickness of supermirror filter
coating_material	NX_CHAR		coating material of supermirror filter
substrate_roughness	NX_FLOAT	NX_LENGTH	substrate roughness (RMS) of supermirror filter
coating_roughness	NX_FLOAT	NX_LENGTH	coating roughness (RMS) of supermirror filter Dimensions: size="1" • dim: incr="1" value="nsurf" value=""

A.2.18 NXflipper

category base (base class)

NXDL source: NXflipper (http://svn.nexusformat.org/definitions/trunk/base_classes/NXflipper.nxdl.xml)

version 1.0

SVN Id Id: NXflipper.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: no included classes

documentation Template of a beamline spin flipper.

Table A.21: Tabular representation of NXflipper[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
type	coil | current-sheet		
flip_turns	NX_FLOAT	NX_PER_LENGTH	Linear density of turns (such as number of turns/cm) in flipping field coils

Table A.21: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
comp_turns	NX_FLOAT	NX_PER_LENGTH	Linear density of turns (such as number of turns/cm) in compensating field coils
guide_turns	NX_FLOAT	NX_PER_LENGTH	Linear density of turns (such as number of turns/cm) in guide field coils
flip_current	NX_FLOAT	NX_CURRENT	Flipping field coil current in "on" state"
comp_current	NX_FLOAT	NX_CURRENT	Compensating field coil current in "on" state"
guide_current	NX_FLOAT	NX_CURRENT	Guide field coil current in "on" state"
thickness	NX_FLOAT	NX_LENGTH	thickness along path of neutron travel

A.2.19 NXgeometry

category `base` (base class)

NXDL source: NXgeometry (http://svn.nexusformat.org/definitions/trunk/base_classes/NXgeometry.nxdl.xml)

version 1.0

SVN Id Id: NXgeometry.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXorientation, NXshape, NXtranslation

documentation This is the description for a general position of a component. It is recommended to name an instance of NXgeometry as "geometry" to aid in the use of the definition in simulation codes such as McStas. Also, in HDF, linked items must share the same name. However, it might not be possible or practical in all situations.

Table A.22: Tabular representation of NXgeometry[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
	NXshape		shape/size information of component
	NXtranslation		translation of component
	NXorientation		orientation of component
description	NX_CHAR		Optional description/label. Probably only present if we are an additional reference point for components rather than the location of a real component
component_index	NX_INT		Position of the component along the beam path. The sample is at 0, components upstream have negative component_index, components downstream have positive component_index.

A.2.20 NXguide

category `base` (base class)

NXDL source: NXguide (http://svn.nexusformat.org/definitions/trunk/base_classes/NXguide.nxdl.xml)

version 1.0

SVN Id Id: NXguide.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXgeometry

documentation Template of NXguide

Table A.23: Tabular representation of NXguide[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
	NXgeometry		
description	NX_CHAR		
incident_angle	NX_FLOAT	NX_ANGLE	
reflectivity	NXdata		Reflectivity as function of wavelength [nsurf,i]
bend_angle_x	NX_FLOAT	NX_ANGLE	
bend_angle_y	NX_FLOAT	NX_ANGLE	
interior_atmosphere	vacuum | helium | argon		
external_material	NX_CHAR		external material outside substrate
m_value	NX_FLOAT		The m value for a supermirror, which defines the supermirror regime in multiples of the critical angle of Nickel. Dimensions: size="1" • dim: index="1" value="nsurf" value=""
substrate_material	NX_FLOAT		Dimensions: size="1" • dim: index="1" value="nsurf" value=""
substrate_thickness	NX_FLOAT	NX_LENGTH	Dimensions: size="1" • dim: index="1" value="nsurf" value=""
coating_material	NX_FLOAT		Dimensions: size="1" • dim: index="1" value="nsurf" value=""
substrate_roughness	NX_FLOAT	NX_LENGTH	Dimensions: size="1" • dim: index="1" value="nsurf" value=""
coating_roughness	NX_FLOAT	NX_LENGTH	Dimensions: size="1" • dim: index="1" value="nsurf" value=""
number_sections	NX_INT	NX_UNITLESS	number of substrate sections

A.2.21 NXinsertion_device

category base (base class)

NXDL source: NXinsertion_device (http://svn.nexusformat.org/definitions/trunk/base_classes/NXinsertion_device.nxdl.xml)

version 1.0

SVN Id Id: NXinsertion_device.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXgeometry

documentation This is the description for an insertion device.

Table A.24: Tabular representation of NXinsertion_device[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
type	undulator | wiggler		
gap	NX_FLOAT	NX_LENGTH	
taper	NX_FLOAT	NX_LENGTH	
phase	NX_FLOAT	NX_ANGLE	
poles	NX_INT	NX_UNITLESS	number of poles
length	NX_FLOAT	NX_LENGTH	length of insertion device
power	NX_FLOAT	NX_POWER	total power delivered by insertion device
energy	NX_FLOAT	NX_ENERGY	energy of peak
bandwidth	NX_FLOAT	NX_ENERGY	bandwidth of peak energy
harmonic	NX_INT	NX_UNITLESS	harmonic of peak
spectrum	NXdata		spectrum of insertion device
	NXgeometry		"Engineering" position of insertion device

A.2.22 NXinstrument

category `base` (base class)

NXDL source: NXinstrument (http://svn.nexusformat.org/definitions/trunk/base_classes/NXinstrument.nxdl.xml)

version 1.0

SVN Id Id: NXinstrument.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXaperture, NXattenuator, NXbeam, NXbeam_stop, NXbending_magnet, NXcollimator, NXcrystal, NXdetector, NXdisk_chopper, NXfermi_chopper, NXfilter, NXflipper, NXguide, NXinsertion_device, NXmirror, NX-moderator, NXpolarizer, NXsource, NXvelocity_selector

documentation Template of instrument descriptions comprising various beamline components. Each component will also be a NeXus group defined by its distance from the sample. Negative distances represent beamline components that are before the sample while positive distances represent components that are after the sample. This device allows the unique identification of beamline components in a way that is valid for both reactor and pulsed instrumentation.

Table A.25: Tabular representation of NXinstrument[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
name @short_name	NX_CHAR NX_CHAR		Name of instrument short name for instrument, perhaps the acronym
	NXaperture		
	NXattenuator		
	NXbeam		

Table A.25: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
	NXbeam_stop		
	NXbending_magnet		
	NXcollimator		
	NXcrystal		
	NXdetector		
	NXdisk_chopper		
	NXfermi_chopper		
	NXfilter		
	NXflipper		
	NXguide		
	NXinsertion_device		
	NXmirror		
	NXmoderator		
	NXpolarizer		
	NXsource		
	NXvelocity_selector		

A.2.23 NXlog

category `base` (base class)

NXDL source: NXlog (http://svn.nexusformat.org/definitions/trunk/base_classes/NXlog.nxdl.xml)

version 1.0

SVN Id Id: NXlog.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: no included classes

documentation Definition of logged information, i.e. information monitored during the run. They contain the logged values and the times at which they were measured as elapsed time since a starting time recorded in ISO8601 format. This method of storing logged data helps to distinguish instances in which a variable is a dimension scale of the data, in which case it is stored in an NXdata group, and instances in which it is logged during the run, when it should be stored in an NXlog group.

Table A.26: Tabular representation of NXlog[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
`time`	NX_FLOAT	NX_TIME	Time of logged entry. The times are relative to the "start" attribute and in the units specified in the "units" attribute.
`@start`	NX_DATE_TIME		
`value`	NX_NUMBER	NX_ANY	Array of logged value, such as temperature
`raw_value`	NX_NUMBER	NX_ANY	Array of raw information, such as thermocouple voltage
`description`	NX_CHAR		Description of logged value
`average_value`	NX_FLOAT	NX_ANY	
`average_value_error`	NX_FLOAT	NX_ANY	standard deviation of average_value
`minimum_value`	NX_FLOAT	NX_ANY	
`maximum_value`	NX_FLOAT	NX_ANY	
`duration`	NX_FLOAT	NX_ANY	Total time log was taken

A.2.24 NXmirror

category `base` (base class)

NXDL source: NXmirror (http://svn.nexusformat.org/definitions/trunk/base_classes/NXmirror.nxdl.xml)

version 1.0

SVN Id Id: NXmirror.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXgeometry

documentation Template of a beamline supermirror.

Table A.27: Tabular representation of NXmirror[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
	NXgeometry		
`description`	NX_CHAR		
`incident_angle`	NX_FLOAT	NX_ANGLE	
`reflectivity`	NXdata		Reflectivity as function of wavelength
`bend_angle_x`	NX_FLOAT	NX_ANGLE	
`bend_angle_y`	NX_FLOAT	NX_ANGLE	
`interior_atmosphere`	vacuum | helium | argon		
`external_material`	NX_CHAR		external material outside substrate
`m_value`	NX_FLOAT	NX_UNITLESS	The m value for a supermirror, which defines the supermirror regime in multiples of the critical angle of Nickel.
`substrate_material`	NX_CHAR		
`substrate_thickness`	NX_FLOAT	NX_LENGTH	
`coating_material`	NX_CHAR		
`substrate_roughness`	NX_FLOAT	NX_LENGTH	
`coating_roughness`	NX_FLOAT	NX_LENGTH	

A.2.25 NXmoderator

category `base` (base class)

NXDL source: NXmoderator (http://svn.nexusformat.org/definitions/trunk/base_classes/NXmoderator.nxdl.xml)

version 1.0

SVN Id Id: NXmoderator.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXgeometry, NXlog

documentation This is the description for a general moderator

Table A.28: Tabular representation of NXmoderator[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
	NXgeometry		"Engineering" position of moderator
distance	NX_FLOAT	NX_LENGTH	Effective distance as seen by measuring radiation
type	H20 | D20 | "Liquid H2" | "Liquid CH4" | "Liquid D2" | "Solid D2" | C | "Solid CH4" | "Solid H2"		
poison_depth	NX_FLOAT	NX_LENGTH	
coupled	NX_BOOLEAN		whether the moderator is coupled
coupling_material	NX_CHAR		The material used for coupling. Usually Cd.
poison_material	Gd | Cd		
temperature	NX_FLOAT	NX_TEMPERATURE	average/nominal moderator temperature
temperature_log	NXlog		log file of moderator temperature
pulse_shape	NXdata		moderator pulse shape

A.2.26 NXmonitor

category base (base class)

NXDL source: NXmonitor (http://svn.nexusformat.org/definitions/trunk/base_classes/NXmonitor.nxdl.xml)

version 1.0

SVN Id Id: NXmonitor.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXgeometry, NXlog

documentation Template of monitor data. It is similar to the NXdata groups containing monitor data and its associated dimension scale, e.g. time_of_flight or wavelength in pulsed neutron instruments. However, it may also include integrals, or scalar monitor counts, which are often used in both in both pulsed and steady-state instrumentation.

Table A.29: Tabular representation of NXmonitor[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
mode	monitor | timer		Count to a preset value based on either clock time (timer) or received monitor counts (monitor).
preset	NX_NUMBER	NX_ANY	preset value for time or monitor
distance	NX_FLOAT	NX_LENGTH	Distance of monitor from sample
range	NX_FLOAT	NX_ANY	Range (X-axis, Time-of-flight, etc.) over which the integral was calculated Dimensions: apparent size="1" • dim: value="2" value=""
integral	NX_NUMBER	NX_ANY	Total integral monitor counts
integral_log	NXlog		Time variation of monitor counts
type	"Fission Chamber" | Scintillator		
time_of_flight	NX_FLOAT	NX_TIME_OF_FLIGHT	Time-of-flight Dimensions: apparent size="1" • dim: ref="efficiency" value=""
efficiency	NX_NUMBER	NX_DIMENSIONLESS	Monitor efficiency Dimensions: apparent size="1" • dim: ref="i" value=""
data @signal @axes	NX_NUMBER NX_INT NX_CHAR	NX_ANY	Monitor data The signal and axes attributes take the same definitions as in NXdata: signal: signal="1" means this is the plottable data axes: axes="names" where names are defined as a comma-delimited string within this attribute in the C-order of the data array Dimensions: apparent size="1" • dim: value=""
sampled_fracti-on	NX_FLOAT	NX_DIMENSIONLESS	Proportion of incident beam sampled by the monitor (0<x<1)
	NXgeometry		Geometry of the monitor
count_time	NX_FLOAT	NX_TIME	Elapsed actual counting time, can be an array of size np when scanning. This is not the difference of the calendar time but the time the instrument was really counting, without pauses or times lost due beam unavailability

A.2.27 NXmonochromator

category base (base class)

NXDL source: NXmonochromator (http://svn.nexusformat.org/definitions/trunk/base_classes/NXmonochromator.nxdl.xml)

version 1.0

SVN Id Id: NXmonochromator.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXgeometry

documentation This is a base class for everything which selects a wavelength or energy, be it a monochromator crystal, a velocity selector, a undulator or whatever expected units wavelength: angstrom energy: eV

Table A.30: Tabular representation of NXmonochromator[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
`wavelength`	NX_FLOAT	NX_WAVELENGTH	wavelength selected
`wavelength_error`	NX_FLOAT	NX_WAVELENGTH	wavelength standard deviation
`energy`	NX_FLOAT	NX_ENERGY	energy selected
`energy_error`	NX_FLOAT	NX_ENERGY	energy standard deviation
`distribution`	NXdata		
`geometry`	NXgeometry		

A.2.28 NXnote

category `base` (base class)

NXDL source: NXnote (http://svn.nexusformat.org/definitions/trunk/base_classes/NXnote.nxdl.xml)

version 1.0

SVN Id Id: NXnote.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: no included classes

documentation This class can be used to store additional information in a NeXus file e.g. pictures, movies, audio, additional text logs

Table A.31: Tabular representation of NXnote[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
`author`	NX_CHAR		Author or creator of note
`date`	NX_DATE_TIME		Date note created/added
`type`	NX_CHAR		Mime content type of note data field e.g. image/jpeg, text/plain, text/html}
`file_name`	NX_CHAR		Name of original file name if note was read from an external source
`description`	NX_CHAR		Title of an image or other details of the note
`data`	NX_BINARY		Binary note data - if text, line terminator is [CR][LF].

A.2.29 NXobject

category `base` (base class)

NXDL source: NXobject (http://svn.nexusformat.org/definitions/trunk/base_classes/NXobject.nxdl.xml)

version 1.0

SVN Id Id: NXuser.nxdl.xml 303 2009-01-30 17:55:59Z Freddie Akeroyd

NeXus Definition Language NXDL

extends class:

other classes included: no included classes

documentation This is the base object of NeXus

No fields or groups are defined (nothing to show in a table at this point).

A.2.30 NXorientation

category `base` (base class)

NXDL source: NXorientation (http://svn.nexusformat.org/definitions/trunk/base_classes/NXorientation.nxdl.xml)

version 1.0

SVN Id Id: NXorientation.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXgeometry

documentation This is the description for a general orientation of a component - it is used by the NXgeometry class

Table A.32: Tabular representation of NXorientation[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
	NXgeometry		Link to another object if we are using relative positioning, else absent
`value`	NX_FLOAT	NX_UNITLESS	The orientation information is stored as direction cosines. The direction cosines will be between the local coordinate directions and the reference directions (to origin or relative NXgeometry). Calling the local unit vectors (x',y',z') and the reference unit vectors (x,y,z) the six numbers will be [x' dot x, x' dot y, x' dot z, y' dot x, y' dot y, y' dot z] where "dot" is the scalar dot product (cosine of the angle between the unit vectors). The unit vectors in both the local and reference coordinates are right-handed and orthonormal. Dimensions: apparent size="2" • dim: value="numobj" value="" • dim: value="6" value=""

A.2.31 NXparameters

category `base` (base class)

NXDL source: NXparameters (http://svn.nexusformat.org/definitions/trunk/base_classes/NXparameters.nxdl.xml)

version 1.0

SVN Id Id: NXparameters.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: no included classes

documentation Container for parameters, usually used in processing or analysis.

Table A.33: Tabular representation of NXparameters[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
`term`	NX_CHAR		A parameter (also known as a term) that is used in or results from processing. Occurences: 0 : 99999999999999999999999
`@units`	NX_CHAR		

A.2.32 NXpolarizer

category `base` (base class)

NXDL source: NXpolarizer (http://svn.nexusformat.org/definitions/trunk/base_classes/NXpolarizer.nxdl.xml)

version 1.0

SVN Id Id: NXpolarizer.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: no included classes

documentation Template of a beamline spin polarizer. This is a first cut. This is NOT the last word.

Table A.34: Tabular representation of NXpolarizer[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
`type`	NX_CHAR		crystal,supermirror
`composition`	NX_CHAR		description of the composition of the polarizing material
`reflection`	NX_INT	NX_UNITLESS	[hkl] values of nominal reflection Dimensions: apparent size="1" • dim: value="3" value=""
`efficiency`	NX_FLOAT	NX_DIMENSIONLESS	polarizing efficiency

A.2.33 NXpositioner

category `base` (base class)

NXDL source: NXpositioner (http://svn.nexusformat.org/definitions/trunk/base_classes/NXpositioner.nxdl.xml)

version 1.0

SVN Id Id: NXpositioner.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: no included classes

documentation This is the description for a generic positioner.

Table A.35: Tabular representation of NXpositioner[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
description	NX_CHAR		description of positioner
value	NX_FLOAT	NX_ANY	best known value of positioner - need [n] as may be scanned Dimensions: size="1" • dim: index="1" value="n" value=""
raw_value	NX_NUMBER	NX_ANY	raw value of positioner - need [n] as may be scanned Dimensions: size="1" • dim: index="1" value="n" value=""
target_value	NX_FLOAT	NX_ANY	targeted (commanded) value of positioner - need [n] as may be scanned Dimensions: size="1" • dim: index="1" value="n" value=""
tolerance	NX_FLOAT	NX_ANY	maximum allowable difference between target_value and value Dimensions: size="1" • dim: index="1" value="n" value=""
soft_limit_min	NX_FLOAT	NX_ANY	minimum allowed limit to set value
soft_limit_max	NX_FLOAT	NX_ANY	maximum allowed limit to set value
controller_rec-ord	NX_CHAR		Hardware device record, e.g. EPICS process variable, taco/tango ...

A.2.34 NXprocess

category base (base class)

NXDL source: NXprocess (http://svn.nexusformat.org/definitions/trunk/base_classes/NXprocess.nxdl.xml)

version 1.0

SVN Id Id: NXprocess.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXnote

documentation Template for a process.

Table A.36: Tabular representation of NXprocess[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
program	NX_CHAR		Name of the program used for reconstruction
version	NX_CHAR		Version of the program used
date	NX_DATE_TIME		Date and time of reconstruction processing.
	NXnote		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. The name will be numbered to allow for ordering of steps.

A.2.35 NXroot

category base (base class)

NXDL source: NXroot (http://svn.nexusformat.org/definitions/trunk/base_classes/NXroot.nxdl.xml)

version 1.0

SVN Id Id: NXroot.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXentry

documentation Definition of the root NeXus group.

Table A.37: Tabular representation of NXroot[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
	NXentry		entries Occurences: 1 : *default*

A.2.36 NXsample

category base (base class)

NXDL source: NXsample (http://svn.nexusformat.org/definitions/trunk/base_classes/NXsample.nxdl.xml)

version 1.0

SVN Id Id: NXsample.nxdl.xml 473 2010-01-26 22:56:14Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXbeam, NXdata, NXenvironment, NXgeometry, NXlog

documentation Template of the state of the sample. This could include scanned variables that are associated with one of the data dimensions, e.g. the magnetic field, or logged data, e.g. monitored temperature vs elapsed time.

Table A.38: Tabular representation of NXsample[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		Descriptive name of sample
chemical_formula	NX_CHAR		The chemical formula specified using CIF conventions. Abbreviated version of CIF standard: • Only recognized element symbols may be used. • Each element symbol is followed by a 'count' number. A count of '1' may be omitted. • A space or parenthesis must separate each cluster of (element symbol + count). • Where a group of elements is enclosed in parentheses, the multiplier for the group must follow the closing parentheses. That is, all element and group multipliers are assumed to be printed as subscripted numbers. • Unless the elements are ordered in a manner that corresponds to their chemical structure, the order of the elements within any group or moiety depends on whether or not carbon is present. • If carbon is present, the order should be: C, then H, then the other elements in alphabetical order of their symbol. If carbon is not present, the elements are listed purely in alphabetic order of their symbol. • This is the 'Hill' system used by Chemical Abstracts.
temperature	NX_FLOAT	NX_TEMPERATURE	Sample temperature. This could be a scanned variable Dimensions: apparent size="1" • dim: value=""
electric_field	NX_FLOAT	NX_VOLTAGE	Applied electric field Dimensions: apparent size="1" • dim: value=""
@direction	NX_CHAR		
magnetic_field	NX_FLOAT	NX_ANY	Applied magnetic field Dimensions: apparent size="1" • dim: value=""
@direction	NX_CHAR		
stress_field	NX_FLOAT	NX_ANY	Applied external stress field Dimensions: apparent size="1" • dim: value=""
@direction	NX_CHAR		
pressure	NX_FLOAT	NX_PRESSURE	Applied pressure Dimensions: apparent size="1" • dim: value=""

Table A.38: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
changer_position	NX_INT	NX_UNITLESS	Sample changer position
unit_cell	NX_FLOAT	NX_LENGTH	Unit cell parameters (lengths and angles) Dimensions: size="2" • dim: index="1" value="n_comp" value="" • dim: index="2" value="6" value=""
unit_cell_volume	NX_FLOAT	NX_VOLUME	Volume of the unit cell Dimensions: size="1" • dim: index="1" value="n_comp" value=""
sample_orientation	NX_FLOAT	NX_ANGLE	This will follow the Busing and Levy convention from Acta.Crysta v22, p457 (1967) Dimensions: size="1" • dim: index="1" value="3" value=""
orientation_matrix	NX_FLOAT		Orientation matrix of single crystal sample. This will follow the Busing and Levy convention from Acta.Crysta v22, p457 (1967) Dimensions: size="3" • dim: index="1" value="n_comp" value="" • dim: index="2" value="3" value="" • dim: index="3" value="3" value=""
mass	NX_FLOAT	NX_MASS	Mass of sample Dimensions: size="1" • dim: index="1" value="n_comp" value=""
density	NX_FLOAT	NX_MASS_DENSITY	Density of sample Dimensions: size="1" • dim: index="1" value="n_comp" value=""
relative_molecular_mass	NX_FLOAT	NX_MASS	Relative Molecular Mass of sample Dimensions: size="1" • dim: index="1" value="n_comp" value=""
type	sample | sample+can | can | "calibration sample" | "normalisation sample" | "simulated data" | none | "sample environment"		
situation	air | vacuum | "inert atmosphere" | "oxidising atmosphere" | "reducing atmosphere" | "sealed can" | other		The atmosphere will be one of the components, which is where its details will be stored; the relevant components will be indicated by the entry in the sample_component member.

Table A.38: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
description	NX_CHAR		Description of the sample
preparation_da-te	NX_DATE_TIME		Date of preparation of the sample
geometry	NXgeometry		The position and orientation of the center of mass of the sample
	NXbeam		Details of beam incident on sample - used to calculate sample/beam interaction point
component	NX_CHAR		Details of the component of the sample and/or can Dimensions: size="1" • dim: index="1" value="n_comp" value=""
sample_compone-nt	sample | can | atmosphere | kit		What type of component we are: "sample | can | atmosphere | kit" Dimensions: size="1" • dim: index="1" value="n_comp" value=""
concentration	NX_FLOAT	NX_MASS_DENSITY	Concentration of each component Dimensions: size="1" • dim: index="1" value="n_comp" value=""
volume_fraction	NX_FLOAT		Volume fraction of each component Dimensions: size="1" • dim: index="1" value="n_comp" value=""
scattering_len-gth_density	NX_FLOAT	NX_SCATTERING_LENGTH_DENSITY	Scattering length density of each component Dimensions: size="1" • dim: index="1" value="n_comp" value=""
unit_cell_class	cubic | tetragonal | orthorhombic | monoclinic | triclinic		In case it is all we know and we want to record/document it Dimensions: size="1" • dim: index="1" value="n_comp" value=""
unit_cell_group	NX_CHAR		Crystallographic point or space group Dimensions: size="1" • dim: index="1" value="n_comp" value=""
path_length	NX_FLOAT	NX_LENGTH	Path length through sample/can for simple case when it does not vary with scattering direction
path_length_wi-ndow	NX_FLOAT	NX_LENGTH	Thickness of a beam entry/exit window on the can (mm) - assumed same for entry and exit
thickness	NX_FLOAT	NX_LENGTH	sample thickness
transmission	NXdata		As a function of Wavelength
temperature_log	NXlog		temperature_log.value is a link to e.g. temperature_env.sensor1.value_log.value
temperature_env	NXenvironment		Additional sample temperature environment information
magnetic_field-_log	NXlog		magnetic_field_log.value is a link to e.g. magnetic_field_env.sensor1.value_log.value
magnetic_field-_env	NXenvironment		Additional sample magnetic environment information
external_DAC	NX_FLOAT	NX_ANY	value sent to user's sample setup

Table A.38: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
external_ADC	NXlog		logged value (or logic state) read from user's setup
short_title	NX_CHAR		20 character fixed length sample description for legends
rotation_angle	NX_FLOAT	NX_ANGLE	Optional rotation angle for the case when the powder diagram has been obtained through an omega-2theta scan like from a traditional single detector powder diffractometer

A.2.37 NXsensor

category base (base class)

NXDL source: NXsensor (http://svn.nexusformat.org/definitions/trunk/base_classes/NXsensor.nxdl.xml)

version 1.0

SVN Id Id: NXsensor.nxdl.xml 473 2010-01-26 22:56:14Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXgeometry, NXlog, NXorientation

documentation This class describes a sensor used to monitor an external condition - the condition itself is described in NXenvironment

Table A.39: Tabular representation of NXsensor[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
model	NX_CHAR		Sensor identification code/model number
name	NX_CHAR		Name for the sensor
short_name	NX_CHAR		Short name of sensor used e.g. on monitor display program
attached_to	NX_CHAR		where sensor is attached to ("sample" | "can")
geometry	NXgeometry		Defines the axes for logged vector quantities if they are not the global instrument axes
measurement	temperature | pH | magnetic_field | "electric field" | conductivity | resistance | voltage | pressure | flow | stress | strain | shear | surface_pressure		name for measured signal

Table A.39: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
type	NX_CHAR		The type of hardware used for the measurement. Examples (suggestions but not restrictions): **Temperature** J | K | T | E | R | S | Pt100 | Rh/Fe **pH** Hg/Hg2Cl2 | Ag/AgCl | ISFET **Ion selective electrode** specify species; e.g. Ca2+ **Magnetic field** Hall **Surface pressure** wilhelmy plate
run_control	NX_BOOLEAN		Is data collection controlled/synchronised to this quantity: 1=no, 0=to "value", 1=to "value_deriv1" etc.
high_trip_value	NX_FLOAT	NX_ANY	Upper control bound of sensor reading if using run_control
low_trip_value	NX_FLOAT	NX_ANY	Lower control bound of sensor reading if using run_control
value	NX_FLOAT	NX_ANY	nominal setpoint or average value - need [n] as may be a vector Dimensions: apparent size="1" • dim: value="n" value=""
value_deriv1	NX_FLOAT	NX_ANY	Nominal/average first derivative of value e.g. strain rate - same dimensions as "value" (may be a vector) Dimensions: apparent size="1" • dim: ref="value" value=""
value_deriv2	NX_FLOAT	NX_ANY	Nominal/average second derivative of value - same dimensions as "value" (may be a vector) Dimensions: apparent size="1" • dim: ref="value" value=""
value_log	NXlog		Time history of sensor readings
value_deriv1_l-og	NXlog		Time history of first derivative of sensor readings
value_deriv2_l-og	NXlog		Time history of second derivative of sensor readings
external_field-_brief	"along beam" | "across beam" | transverse | solenoidal | "flow shear gradient" | "flow vorticity"		
external_field-_full	NXorientation		For complex external fields not satisfied by External_field_brief

A.2.38 NXshape

category base (base class)

NXDL source: NXshape (http://svn.nexusformat.org/definitions/trunk/base_classes/NXshape.nxdl.xml)

version 1.0

SVN Id Id: NXshape.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: no included classes

documentation This is the description of the general shape and size of a component, which may be made up of "numobj" separate elements - it is used by the NXgeometry class

Table A.40: Tabular representation of NXshape[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
`shape`	nxcylinder | nxbox | nxsphere | nxcone		general shape of a component
`size`	NX_FLOAT	NX_LENGTH	physical extent of the object along its local axes (after NXorientation) with the center of mass at the local origin (after NXtranslation). The meaning and location of these axes will vary according to the value of the "shape" variable. nshapepar defines how many parameters. For the "nxcylinder" type the parameters are (diameter,height). For the "nxbox" type the parameters are (length,width,height). For the "nxsphere" type the parameters are (diameter). Dimensions: apparent size="2" • dim: value="numobj" value="" • dim: value="nshapepar" value=""

A.2.39 NXsource

category `base` (base class)

NXDL source: NXsource (http://svn.nexusformat.org/definitions/trunk/base_classes/NXsource.nxdl.xml)

version 1.0

SVN Id Id: NXsource.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXgeometry, NXnote

documentation Template of the neutron or x-ray source, insertion devices and/or moderators.

Table A.41: Tabular representation of NXsource[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
distance	NX_FLOAT	NX_LENGTH	Effective distance from sample Distance as seen by radiation from sample. This number should be negative to signify that it is upstream of the sample.
name @short_name	NX_CHAR NX_CHAR		Name of source short name for source, perhaps the acronym
type	"Spallation Neutron Source" | "Pulsed Reactor Neutron Source" | "Reactor Neutron Source" | "Synchrotron X-ray Source" | "Pulsed Muon Source" | "Rotating Anode X-ray" | "Fixed Tube X-ray" | "UV Laser" | "Free-Electron Laser" | "Optical Laser"		type of radiation source
probe	neutron | x-ray | muon | electron | ultraviolet | "visible light"		type of radiation probe
power	NX_FLOAT	NX_POWER	Source power
current	NX_FLOAT	NX_CURRENT	Accelerator current
voltage	NX_FLOAT	NX_VOLTAGE	Accelerator voltage
frequency	NX_FLOAT	NX_FREQUENCY	Frequency of pulsed source
period	NX_FLOAT	NX_PERIOD	Period of pulsed source
target_material	Ta | W | depleted_U | enriched_U | Hg | Pb | C		Pulsed source target material
notes	NXnote		any source/facility related messages/events that occurred during the experiment
pulse_width	NX_FLOAT	NX_TIME	width of source pulse
pulse_shape	NXdata		source pulse shape
mode	"Single Bunch" | "Multi Bunch"		source operating mode
top_up	NX_BOOLEAN		Is the synchrotron operating in top_up mode?
geometry	NXgeometry		"Engineering" location of source

A.2.40 NXtranslation

category base (base class)

NXDL source: NXtranslation (http://svn.nexusformat.org/definitions/trunk/base_classes/NXtranslation.nxdl.xml)

version 1.0

SVN Id Id: NXtranslation.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXgeometry

documentation This is the description for the general spatial location of a component - it is used by the NXgeometry class

Table A.42: Tabular representation of NXtranslation[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
geometry	NXgeometry		Link to other object if we are relative , else absent
distances	NX_FLOAT	NX_LENGTH	(x,y,z)}{This field and the angle field describe the position of a component. For absolute position, the origin is the scattering center (where a perfectly aligned sample would be) with the z-axis pointing downstream and the y-axis pointing gravitationally up. For a relative position the NXtranslation is taken into account before the NXorientation. The axes are right-handed and orthonormal. Dimensions: apparent size="2" • dim: value="numobj" value="" • dim: value="3" value=""

A.2.41 NXuser

category base (base class)

NXDL source: NXuser (http://svn.nexusformat.org/definitions/trunk/base_classes/NXuser.nxdl.xml)

version 1.0

SVN Id Id: NXuser.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: no included classes

documentation Template of user's contact information. The format allows more than one user with the same affiliation and contact information, but a second NXuser group should be used if they have different affiliations, etc.

Table A.43: Tabular representation of NXuser[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		Name of user responsible for this entry

Table A.43: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
role	NX_CHAR		Role of user responsible for this entry. Suggested roles are "local_contact", "principal_investigator", and "proposer"
affiliation	NX_CHAR		Affiliation of user
address	NX_CHAR		Address of user
telephone_numb-er	NX_CHAR		Telephone number of user
fax_number	NX_CHAR		Fax number of user
email	NX_CHAR		Email of user
facility_user_-id	NX_CHAR		facility based unique identifier for this person e.g. their identification code on the facility address/contact database

A.2.42 NXvelocity_selector

category base (base class)

NXDL source: NXvelocity_selector (http://svn.nexusformat.org/definitions/trunk/base_classes/NXvelocity_selector.nxdl.xml)

version 1.0

SVN Id Id: NXvelocity_selector.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXgeometry

documentation This is the description for a (typically neutron) velocity selector

Table A.44: Tabular representation of NXvelocity_selector[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
type	NX_CHAR		velocity selector type
rotation_speed	NX_FLOAT	NX_FREQUENCY	velocity selector rotation speed
radius	NX_FLOAT	NX_LENGTH	radius at beam centre
spwidth	NX_FLOAT	NX_LENGTH	spoke width at beam centre
length	NX_FLOAT	NX_LENGTH	rotor length
num	NX_INT	NX_UNITLESS	number of spokes/lamella
twist	NX_FLOAT	NX_ANGLE	twist angle along axis
table	NX_FLOAT	NX_ANGLE	offset vertical angle
height	NX_FLOAT	NX_LENGTH	input beam height
width	NX_FLOAT	NX_LENGTH	input beam width
wavelength	NX_FLOAT	NX_WAVELENGTH	wavelength
wavelength_spr-ead	NX_FLOAT	NX_WAVELENGTH	deviation FWHM /Wavelength
geometry	NXgeometry		

A.3 NeXus Application Classes

A description of each NeXus Application Class is given.

A.3.1 NXarchive

category `application` (application definition)

NXDL source: NXarchive (http://svn.nexusformat.org/definitions/trunk/applications/NXarchive.nxdl.xml)

version 1.0b

SVN Id Id: NXarchive.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXentry, NXinstrument, NXsample, NXsource, NXuser

documentation This is a definition for data to be archived by ICAT

Table A.45: Tabular representation of NXarchive[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
`entry`	NXentry		Look for special case table **entry[NXentry](within NXarchive[definition])** below.
`@index`	NX_CHAR		

A.3.1.1 Special case table: entry[NXentry](within NXarchive[definition])

Table A.46: Tabular representation of entry[NXentry](within NXarchive[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
`title`	NX_CHAR		
`experiment_identifer`	NX_CHAR		unique identifier for the experiment
`experiment_description`	NX_CHAR		Brief description of the experiment and its objectives
`collection_identifier`	NX_CHAR		ID of user or DAQ define group of data files
`collection_description`	NX_CHAR		Brief summary of the collection, including grouping criteria
`entry_identifier`	NX_CHAR		unique identifier for this measurement as provided by the facility
`start_time`	NX_DATE_TIME		
`end_time`	NX_DATE_TIME		
`duration`	NX_FLOAT	NX_TIME	
`collection_time`	NX_FLOAT	NX_TIME	
`run_cycle`	NX_CHAR		
`revision`	NX_CHAR		revision ID of this file, may be after recalibration, reprocessing etc.

Table A.46: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
definition	NXmonopd		Official NeXus DTD or NXDL schema to which this file conforms
program @version	NX_CHAR NX_CHAR		The program and version used for generating this file
release_date	NX_CHAR	NX_TIME	when this file is to be released into PD
user	NXuser		Look for special case table **user[NXuser](within entry[NXentry])** below.
instrument	NXinstrument		Look for special case table **instrument[NXinstrument](within entry[NXentry])** below.
sample	NXsample		Look for special case table **sample[NXsample](within entry[NXentry])** below.

A.3.1.2 Special case table: user[NXuser](within entry[NXentry])

Table A.47: Tabular representation of user[NXuser](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		
role	NX_CHAR		role of the user
facility_user_-id	NX_CHAR		ID of the user in the facility burocracy database

A.3.1.3 Special case table: instrument[NXinstrument](within entry[NXentry])

Table A.48: Tabular representation of instrument[NXinstrument](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
	NXsource		Look for special case table **[NXsource](within instrument[NXinstrument])** below.
name	NX_CHAR		
description	NX_CHAR		Brief description of the instrument

A.3.1.4 Special case table: [NXsource](within instrument[NXinstrument])

Table A.49: Tabular representation of [NXsource](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
type	"Spallation Neutron Source" | "Pulsed Reactor Neutron Source" | "Reactor Neutron Source" | "Synchrotraon X-Ray Source" | "Pulsed Muon Source" | "Rotating Anode X-Ray" | "Fixed Tube X-Ray"		
name	NX_CHAR		
probe	neutron | x-ray | electron		

A.3.1.5 Special case table: sample[NXsample](within entry[NXentry])

Table A.50: Tabular representation of sample[NXsample](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		Descriptive name of sample
sample_id	NX_CHAR		Unique database id of the sample
description	NX_CHAR		
type	sample | sample+can | "calibration sample" | "normalisation sample" | "simulated data" | none | sample_environment		
chemical_formula	NX_CHAR		Chemical formular fomatted according to CIF conventions
preparation_date	NX_CHAR	NX_TIME	
situation	NX_CHAR		Description of the environment the sample is in: air, vacuum, oxiding atmosphere, dehyrated, etc.
temperature	NX_FLOAT	NX_TEMPERATURE	
magnetic_field	NX_FLOAT	NX_CURRENT	
electric_field	NX_FLOAT	NX_VOLTAGE	
stress_field	NX_FLOAT	NX_UNITLESS	
pressure	NX_FLOAT	NX_PRESSURE	

A.3.2 NXgisas

category `application` (application definition)

NXDL source: NXgisas (http://svn.nexusformat.org/definitions/trunk/applications/NXgisas.nxdl.xml)

version 1.0b

SVN Id

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXmonochromator, NXsample, NXsource

documentation This is an application definition for raw data from a grazing incidence small angle diffractometer GISAS for either x-ray or neutrons

Table A.51: Tabular representation of NXgisas[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
`entry`	NXentry		Look for special case table **entry[NXentry](within NXgisas[definition])** below.

A.3.2.1 Special case table: entry[NXentry](within NXgisas[definition])

Table A.52: Tabular representation of entry[NXentry](within NXgisas[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
`title`	NX_CHAR		
`start_time`	NX_DATE_TIME		
`end_time`	NX_DATE_TIME		
`definition`	NXgisas		Official NeXus DTD or NXDL schema to which this file conforms
`instrument`	NXinstrument		Look for special case table **instrument[NXinstrument](within entry[NXentry])** below.
`sample`	NXsample		Look for special case table **sample[NXsample](within entry[NXentry])** below.
`control`	NXmonitor		Look for special case table **control[NXmonitor](within entry[NXentry])** below.
`data`	NXdata		List of links: **`data`** `/NXentry/NXinstrument/NX-detector/data`

A.3.2.2 Special case table: instrument[NXinstrument](within entry[NXentry])

Table A.53: Tabular representation of instrument[NXinstrument](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
`name`	NX_CHAR		
`source`	NXsource		Look for special case table **source[NXsource](within instrument[NXinstrument])** below.
`monochromator`	NXmonochromator		Look for special case table **monochromator[NXmonochromator](within instrument[NXinstrument])** below.
`detector`	NXdetector		Look for special case table **detector[NXdetector](within instrument[NXinstrument])** below.

A.3.2.3 Special case table: source[NXsource](within instrument[NXinstrument])

Table A.54: Tabular representation of source[NXsource](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
`type`	NX_CHAR		
`name`	NX_CHAR		
`probe`	neutron | x-ray		

A.3.2.4 Special case table: monochromator[NXmonochromator](within instrument[NXinstrument])

Table A.55: Tabular representation of monochromator[NXmonochromator](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
`wavelength`	NX_FLOAT	NX_WAVELENGTH	

A.3.2.5 Special case table: detector[NXdetector](within instrument[NXinstrument])

Table A.56: Tabular representation of detector[NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
`data`	NX_INT		Dimensions: size="2" • dim: index="1" value="xsize" value="" • dim: index="2" value="ysize" value=""
`distance`	NX_FLOAT	NX_LENGTH	
`x_pixel_size`	NX_FLOAT	NX_LENGTH	
`y_pixel_size`	NX_FLOAT	NX_LENGTH	

A.3.2.6 Special case table: sample[NXsample](within entry[NXentry])

Table A.57: Tabular representation of sample[NXsample](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		Descriptive name of sample
polar_angle	NX_FLOAT	NX_ANGLE	In NeXus this is the gazing incidence angle on the sample

A.3.2.7 Special case table: control[NXmonitor](within entry[NXentry])

Table A.58: Tabular representation of control[NXmonitor](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
mode	monitor | timer		Count to a preset value based on either clock time (timer) or received monitor counts (monitor).
preset	NX_FLOAT	NX_ANY	preset value for time or monitor
integral	NX_INT		Total integral monitor counts
time_of_flight	NX_FLOAT	NX_TIME_OF_FLIGHT	Time channels
data	NX_INT		Monitor counts in each time channel

A.3.3 NXiqproc

category `application` (application definition)

NXDL source: NXiqproc (http://svn.nexusformat.org/definitions/trunk/applications/NXiqproc.nxdl.xml)

version 1.0b

SVN Id Id: NXiqproc.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXentry, NXinstrument, NXparameters, NXprocess, NXsample, NXsource

documentation Actually this is a template from which to start an application definition.

Table A.59: Tabular representation of NXiqproc[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
 @entry	NXentry NX_CHAR		Look for special case table **[NXentry](within NXiqproc[definition])** below.

A.3.3.1 Special case table: [NXentry](within NXiqproc[definition])

Table A.60: Tabular representation of [NXentry](within NXiqproc[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
`title`	NX_CHAR		
`definition`	NXiqproc		Official NeXus DTD or NXDL schema to which this file conforms
`instrument`	NXinstrument		Look for special case table **instrument[NXinstrument](within [NXentry])** below.
	NXsample		Look for special case table **[NXsample](within [NXentry])** below.
`reduction`	NXprocess		Look for special case table **reduction[NXprocess](within [NXentry])** below.
	NXdata		Look for special case table **[NXdata](within [NXentry])** below.

A.3.3.2 Special case table: instrument[NXinstrument](within [NXentry])

Table A.61: Tabular representation of instrument[NXinstrument](within [NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
	NXsource		Look for special case table **[NXsource](within instrument[NXinstrument])** below.
`name`	NX_CHAR		Name of the instrument from which this data was reduced.

A.3.3.3 Special case table: [NXsource](within instrument[NXinstrument])

Table A.62: Tabular representation of [NXsource](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
`type`	NX_CHAR		
`name`	NX_CHAR		
`probe`	neutron | x-ray | electron		

A.3.3.4 Special case table: [NXsample](within [NXentry])

Table A.63: Tabular representation of [NXsample](within [NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
`name`	NX_CHAR		Descriptive name of sample

A.3.3.5 Special case table: reduction[NXprocess](within [NXentry])

Table A.64: Tabular representation of reduction[NXprocess](within [NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
`program`	NX_CHAR		
`version`	NX_CHAR		
`input`	NXparameters		Input parameters for the reduction program used Look for special case table **input[NXparameters](within reduction[NXprocess])** below.
`output`	NXparameters		Eventual output parameters from the data reduction program used

A.3.3.6 Special case table: input[NXparameters](within reduction[NXprocess])

Table A.65: Tabular representation of input[NXparameters](within reduction[NXprocess])

Name and Attributes	Type	Units	Description (and Occurrences)
`filenames`	NX_CHAR		Raw data files used to generate this I(Q)

A.3.3.7 Special case table: [NXdata](within [NXentry])

Table A.66: Tabular representation of [NXdata](within [NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
`data`	NX_INT		This is I(Q). The client has to analyse the dimensions of I(Q). Often, multiple I(Q) for various environment conditions are measured; that would be the first dimension. Q can be multidimensional, this accounts for the further dimensions in the data Dimensions: size="3" • dim: index="1" value="NE" value="" • dim: index="2" value="NQX" value="" • dim: index="3" value="NQY" value=""
`variable`	NX_CHAR		Dimensions: size="1" • dim: index="1" value="NE" value=""
`@varied_variable`	NX_CHAR		The real name of the varied variable in the first dim of data, temperature, P, MF etc...
`qx`	NX_CHAR		Values for the first dimension of Q Dimensions: size="1" • dim: index="1" value="NQX" value=""

Table A.66: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
qy	NX_CHAR		Values for the second dimension of Q Dimensions: size="1" • dim: index="1" value="NQY" value=""

A.3.4 NXmonopd

category application (application definition)

NXDL source: NXmonopd (http://svn.nexusformat.org/definitions/trunk/applications/NXmonopd.nxdl.xml)

version 1.0b

SVN Id Id: NXmonopd.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class:

other classes included: NXcrystal, NXdata, NXdetector, NXinstrument, NXmonitor, NXsample, NXsource

documentation Monochromatic Neutron and X-Ray Powder Diffraction. Instrument definition for a powder diffractometer at a monochromatic neutron or X-ray beam. This is both suited for a powder diffractometer with a single detector or a powder diffractometer with a position sensitive detector.

Table A.67: Tabular representation of NXmonopd[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
title	NX_CHAR		
start_time	NX_DATE_TIME		
definition	NXmonopd		Official NeXus DTD or NXDL schema to which this file conforms
	NXinstrument		Look for special case table **[NXinstrument](within NXmonopd[definition])** below.
	NXsample		Look for special case table **[NXsample](within NXmonopd[definition])** below.
	NXmonitor		Look for special case table **[NXmonitor](within NXmonopd[definition])** below.
	NXdata		List of links: **polar_angle** /NXentry/NXinstrument/NXdetector/polar_angle **data** /NXentry/NXinstrument/NXdetector/data

A.3.4.1 Special case table: [NXinstrument](within NXmonopd[definition])

Table A.68: Tabular representation of [NXinstrument](within NXmonopd[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
	NXsource		Look for special case table **[NXsource](within [NXinstrument])** below.
	NXcrystal		Look for special case table **[NXcrystal](within [NXinstrument])** below.
	NXdetector		Look for special case table **[NXdetector](within [NXinstrument])** below.

A.3.4.2 Special case table: [NXsource](within [NXinstrument])

Table A.69: Tabular representation of [NXsource](within [NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
`type`	NX_CHAR		
`name`	NX_CHAR		
`probe`	neutron | x-ray | electron		

A.3.4.3 Special case table: [NXcrystal](within [NXinstrument])

Table A.70: Tabular representation of [NXcrystal](within [NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
`wavelength`	NX_FLOAT	NX_WAVELENGTH	Optimum diffracted wavelength Dimensions: size="1" • dim: index="1" value="i" value=""

A.3.4.4 Special case table: [NXdetector](within [NXinstrument])

Table A.71: Tabular representation of [NXdetector](within [NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
`polar_angle`	NX_FLOAT		where ndet = number of detectors Dimensions: size="1" • dim: index="1" value="ndet" value=""
`data`	NX_INT		detector signal (usually counts) are already corrected for detector efficiency Dimensions: size="1" • dim: index="1" value="ndet" value=""

A.3.4.5 Special case table: [NXsample](within NXmonopd[definition])

Table A.72: Tabular representation of [NXsample](within NXmonopd[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		Descriptive name of sample
rotation_angle	NX_FLOAT	NX_ANGLE	Optional rotation angle for the case when the powder diagram has been obtained through an omega-2theta scan like from a traditional single detector powder diffractometer

A.3.4.6 Special case table: [NXmonitor](within NXmonopd[definition])

Table A.73: Tabular representation of [NXmonitor](within NXmonopd[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
mode	monitor | timer		Count to a preset value based on either clock time (timer) or received monitor counts (monitor).
preset	NX_FLOAT		preset value for time or monitor
integral	NX_FLOAT	NX_ANY	Total integral monitor counts

A.3.5 NXrefscan

category application (application definition)

NXDL source: NXrefscan (http://svn.nexusformat.org/definitions/trunk/applications/NXrefscan.nxdl.xml)

version 1.0b

SVN Id

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXmonochromator, NXsample, NXsource

documentation This is an application definition for a monochromatic scanning reflectometer. It does not have the information to calculate the resolution since it does not have any apertures.

Table A.74: Tabular representation of NXrefscan[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
entry	NXentry		Look for special case table **entry[NXentry](within NXrefscan[definition])** below.

A.3.5.1 Special case table: entry[NXentry](within NXrefscan[definition])

Table A.75: Tabular representation of entry[NXentry](within NXrefscan[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
title	NX_CHAR		
start_time	NX_DATE_TIME		
end_time	NX_DATE_TIME		
definition	NXrefscan		Official NeXus DTD or NXDL schema to which this file conforms
instrument	NXinstrument		Look for special case table **instrument[NXinstrument](within entry[NXentry])** below.
sample	NXsample		Look for special case table **sample[NXsample](within entry[NXentry])** below.
control	NXmonitor		Look for special case table **control[NXmonitor](within entry[NXentry])** below.
data	NXdata		List of links: **data** /NXentry/NXinstrument/NXdetector/data **rotation_angle** /NXentry/NXsample/rotation_angle **polar_angle** /NXentry/NXinstrument/NXdetector/polar_angle

A.3.5.2 Special case table: instrument[NXinstrument](within entry[NXentry])

Table A.76: Tabular representation of instrument[NXinstrument](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
	NXsource		Look for special case table **[NXsource](within instrument[NXinstrument])** below.
monochromator	NXmonochromator		Look for special case table **monochromator[NXmonochromator](within instrument[NXinstrument])** below.
	NXdetector		Look for special case table **[NXdetector](within instrument[NXinstrument])** below.

A.3.5.3 Special case table: [NXsource](within instrument[NXinstrument])

Table A.77: Tabular representation of [NXsource](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
type	NX_CHAR		
name	NX_CHAR		
probe	neutron | x-ray | electron		

A.3.5.4 Special case table: monochromator[NXmonochromator](within instrument[NXinstrument])

Table A.78: Tabular representation of monochromator[NXmonochromator](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
wavelength	NX_FLOAT	NX_WAVELENGTH	

A.3.5.5 Special case table: [NXdetector](within instrument[NXinstrument])

Table A.79: Tabular representation of [NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
data	NX_INT		Dimensions: size="1" • dim: index="1" value="NP" value=""
polar_angle	NX_FLOAT	NX_ANGLE	Dimensions: size="1" • dim: index="1" value="NP" value=""

A.3.5.6 Special case table: sample[NXsample](within entry[NXentry])

Table A.80: Tabular representation of sample[NXsample](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		Descriptive name of sample
rotation_angle	NX_FLOAT	NX_ANGLE	Dimensions: size="1" • dim: index="1" value="NP" value=""

A.3.5.7 Special case table: control[NXmonitor](within entry[NXentry])

Table A.81: Tabular representation of control[NXmonitor](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
mode	monitor | timer		Count to a preset value based on either clock time (timer) or received monitor counts (monitor).
preset	NX_FLOAT		preset value for time or monitor
data	NX_FLOAT	NX_ANY	Monitor counts for each step Dimensions: size="1" • dim: index="1" value="NP" value=""

A.3.6 NXreftof

category `application` (application definition)

NXDL source: NXreftof (http://svn.nexusformat.org/definitions/trunk/applications/NXreftof.nxdl.xml)

version 1.0b

SVN Id

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXdetector, NXdisk_chopper, NXentry, NXinstrument, NXmonitor, NXsample

documentation This is an application definition for raw data from a TOF reflectometer.

Table A.82: Tabular representation of NXreftof[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
`entry`	NXentry		Look for special case table **entry[NXentry](within NXreftof[definition])** below.

A.3.6.1 Special case table: entry[NXentry](within NXreftof[definition])

Table A.83: Tabular representation of entry[NXentry](within NXreftof[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
`title`	NX_CHAR		
`start_time`	NX_DATE_TIME		
`end_time`	NX_DATE_TIME		
`definition`	NXreftof		Official NeXus DTD or NXDL schema to which this file conforms
`instrument`	NXinstrument		Look for special case table **instrument[NXinstrument](within entry[NXentry])** below.
`sample`	NXsample		Look for special case table **sample[NXsample](within entry[NXentry])** below.
`control`	NXmonitor		Look for special case table **control[NXmonitor](within entry[NXentry])** below.
`data`	NXdata		List of links: **`data`** `/NXentry/NXinstrument/NXdetector/data` **`time_binning`** `/NXentry/NXinstrument/NXdetector/time_binning`

A.3.6.2 Special case table: instrument[NXinstrument](within entry[NXentry])

Table A.84: Tabular representation of instrument[NXinstrument](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		
chopper	NXdisk_chopper		Look for special case table **chopper[NXdisk_chopper](within instrument[NXinstrument])** below.
detector	NXdetector		Look for special case table **detector[NXdetector](within instrument[NXinstrument])** below.

A.3.6.3 Special case table: chopper[NXdisk_chopper](within instrument[NXinstrument])

Table A.85: Tabular representation of chopper[NXdisk_chopper](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
distance	NX_FLOAT	NX_LENGTH	Distance between chopper and sample

A.3.6.4 Special case table: detector[NXdetector](within instrument[NXinstrument])

Table A.86: Tabular representation of detector[NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
data	NX_INT		Dimensions: size="3" • dim: index="1" value="xsize" value="" • dim: index="2" value="ysize" value="" • dim: index="3" value="nTOF" value=""
time_of_flight	NX_FLOAT	NX_TIME_OF_FLIGHT	Array of time values for each bin in a time-of-flight measurement Dimensions: size="1" • dim: index="1" value="nTOF" value=""
distance	NX_FLOAT	NX_LENGTH	
polar_angle	NX_FLOAT	NX_ANGLE	
x_pixel_size	NX_FLOAT	NX_LENGTH	
y_pixel_size	NX_FLOAT	NX_LENGTH	

A.3.6.5 Special case table: sample[NXsample](within entry[NXentry])

Table A.87: Tabular representation of sample[NXsample](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		Descriptive name of sample

Table A.87: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
rotation_angle	NX_FLOAT	NX_ANGLE	

A.3.6.6 Special case table: control[NXmonitor](within entry[NXentry])

Table A.88: Tabular representation of control[NXmonitor](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
mode	monitor | timer		Count to a preset value based on either clock time (timer) or received monitor counts (monitor).
preset	NX_FLOAT	NX_ANY	preset value for time or monitor
integral	NX_INT		Total integral monitor counts
time_of_flight	NX_FLOAT	NX_TIME_OF_FLIGHT	Time channels
data	NX_INT		Monitor counts in each time channel

A.3.7 NXsas

category application (application definition)

NXDL source: NXsas (http://svn.nexusformat.org/definitions/trunk/applications/NXsas.nxdl.xml)

version 1.0b

SVN Id

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXcollimator, NXdata, NXdetector, NXentry, NXgeometry, NXinstrument, NXmonitor, NXmonochromator, NXsample, NXshape, NXsource

documentation This is an application definition for 2-D small angle scattering data collected with a monochromatic beam and an area detector. It is meant to be suitable both for neutron SANS and X-ray SAXS data.

Table A.89: Tabular representation of NXsas[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
	NXentry		Look for special case table **[NXentry](within NXsas[definition])** below.
@entry	NX_CHAR		NeXus convention is to use "entry1", "entry2", ... for analysis software to locate each entry

A.3.7.1 Special case table: [NXentry](within NXsas[definition])

Table A.90: Tabular representation of [NXentry](within NXsas[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
`title`	NX_CHAR		
`start_time`	NX_DATE_TIME		
`end_time`	NX_DATE_TIME		
`definition`	NXsas		Official NeXus DTD or NXDL schema to which this file conforms
`instrument`	NXinstrument		Look for special case table **instrument[NXinstrument](within [NXentry])** below.
`sample`	NXsample		Look for special case table **sample[NXsample](within [NXentry])** below.
`control`	NXmonitor		Look for special case table **control[NXmonitor](within [NXentry])** below.
`data`	NXdata		List of links: **`data`** `/NXentry/NXinstrument/NX-detector/data`

A.3.7.2 Special case table: instrument[NXinstrument](within [NXentry])

Table A.91: Tabular representation of instrument[NXinstrument](within [NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
`source`	NXsource		Look for special case table **source[NXsource](within instrument[NXinstrument])** below.
`monochromator`	NXmonochromator		Look for special case table **monochromator[NXmonochromator](within instrument[NXinstrument])** below.
`collimator`	NXcollimator		Look for special case table **collimator[NXcollimator](within instrument[NXinstrument])** below.
`detector`	NXdetector		Look for special case table **detector[NXdetector](within instrument[NXinstrument])** below.
`name`	NX_CHAR		Name of the instrument actually used to perform the experiment

A.3.7.3 Special case table: source[NXsource](within instrument[NXinstrument])

Table A.92: Tabular representation of source[NXsource](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
`type`	NX_CHAR		type of radiation source
`name`	NX_CHAR		Name of the radiation source
`probe`	neutron | x-ray		

A.3.7.4 Special case table: monochromator[NXmonochromator](within instrument[NXinstrument])

Table A.93: Tabular representation of monochromator[NXmonochromator](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
wavelength	NX_FLOAT	NX_WAVELENGTH	The wavelength of the radiation
wavelength_spread	NX_FLOAT		delta_lambda/lambda. The wavelength spread. Important with neutrons to calculate resolution.

A.3.7.5 Special case table: collimator[NXcollimator](within instrument[NXinstrument])

Table A.94: Tabular representation of collimator[NXcollimator](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
geometry	NXgeometry		Look for special case table **geometry[NXgeometry](within collimator[NXcollimator])** below.

A.3.7.6 Special case table: geometry[NXgeometry](within collimator[NXcollimator])

Table A.95: Tabular representation of geometry[NXgeometry](within collimator[NXcollimator])

Name and Attributes	Type	Units	Description (and Occurrences)
shape	NXshape		Look for special case table **shape[NXshape](within geometry[NXgeometry])** below.

A.3.7.7 Special case table: shape[NXshape](within geometry[NXgeometry])

Table A.96: Tabular representation of shape[NXshape](within geometry[NXgeometry])

Name and Attributes	Type	Units	Description (and Occurrences)
shape	nxcylinder | nxbox		
size	NX_FLOAT	NX_LENGTH	The collimation length

A.3.7.8 Special case table: detector[NXdetector](within instrument[NXinstrument])

Table A.97: Tabular representation of detector[NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
data	NX_INT		This is area detector data, of number of x-pixel versus number of y-pixels. Since the beam center is to be determined as a step of data reduction, it is not necessary to document or assume the position of the beam center in acquired data. Dimensions: size="2" • dim: index="1" value="nXPixel" value="" • dim: index="2" value="nYPixel" value=""
distance	NX_FLOAT	NX_LENGTH	The distance between detector and sample
x_pixel_size	NX_FLOAT	NX_LENGTH	Physical size of a pixel in x-direction
y_pixel_size	NX_FLOAT	NX_LENGTH	Size of a pixel in y direction

A.3.7.9 Special case table: sample[NXsample](within [NXentry])

Table A.98: Tabular representation of sample[NXsample](within [NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		Descriptive name of sample
thickness	NX_FLOAT	NX_LENGTH	sample thickness

A.3.7.10 Special case table: control[NXmonitor](within [NXentry])

Table A.99: Tabular representation of control[NXmonitor](within [NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
mode	monitor | timer		Count to a preset value based on either clock time (timer) or received monitor counts (monitor).
preset	NX_FLOAT		preset value for time or monitor
integral	NX_FLOAT	NX_ANY	Total integral monitor counts

A.3.8 NXscan

category `application` (application definition)

NXDL source: NXscan (http://svn.nexusformat.org/definitions/trunk/applications/NXscan.nxdl.xml)

version 1.0b

SVN Id

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample

documentation Application definition for a generic scan instrument. This definition is more an example then a stringent definition as the content of a given NeXus scan file needs to differ for different types of scans. This example definition shows a scan like done on a rotation camera: the sample is rotated and a detector image, the rotation angle and a monitor value is stored at each step in the scan. In the following I use the symbol NP as a placeholder for the number of scan points. These are the rules for storing scan data in NeXus files which are implemented in this example: - Each value varied throughout a scan is stored as an array of length NP at its respective location within the NeXus hierarchy. - For area detectors, NP is the first dimension, example for a detector of 256x256: data[NP,256,256] - The NXdata group contains links to all variables varied in the scan and the data. This to give an equivalent to the more familiar classical tabular representation of scans. These rules are there for a reason: HDF allows the first dimension of a data set to be unlimited. This means the data can be appended too. Thus a NeXus file built according to the rules given above can be used in the following way: - At the start of a scan, write all the static information. - At each scan point append new data from varied variables and the detector to the file.

Table A.100: Tabular representation of NXscan[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
	NXentry		Look for special case table **[NXentry](within NXscan[definition])** below.

A.3.8.1 Special case table: [NXentry](within NXscan[definition])

Table A.101: Tabular representation of [NXentry](within NXscan[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
`title`	NX_CHAR		
`start_time`	NX_DATE_TIME		
`end_time`	NX_DATE_TIME		
`definition`	NX_CHAR		Identifies the NXDL that describes the structure of this data file
`@URL`	NX_CHAR		Uniform Resource Locator for the base_class, application, or contributed NXDL that describes the structure of this data file
	NXinstrument		Look for special case table **[NXinstrument](within [NXentry])** below.
	NXsample		Look for special case table **[NXsample](within [NXentry])** below.
	NXmonitor		Look for special case table **[NXmonitor](within [NXentry])** below.
	NXdata		List of links: **`data`** `/NXentry/NXinstrument/NXdetector/data` **`rotation_angle`** `/NXentry/NXsample/rotation_angle`

A.3.8.2 Special case table: [NXinstrument](within [NXentry])

Table A.102: Tabular representation of [NXinstrument](within [NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
	NXdetector		Look for special case table **[NXdetector](within [NXinstrument])** below.

A.3.8.3 Special case table: [NXdetector](within [NXinstrument])

Table A.103: Tabular representation of [NXdetector](within [NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
`data`	NX_INT		Dimensions: size="3" • dim: index="1" value="NP" value="" • dim: index="2" value="xdim" value="" • dim: index="3" value="ydim" value=""

A.3.8.4 Special case table: [NXsample](within [NXentry])

Table A.104: Tabular representation of [NXsample](within [NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
`rotation_angle`	NX_FLOAT		Dimensions: size="1" • dim: index="1" value="NP" value=""

A.3.8.5 Special case table: [NXmonitor](within [NXentry])

Table A.105: Tabular representation of [NXmonitor](within [NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
`data`	NX_INT		Dimensions: size="1" • dim: index="1" value="NP" value=""

A.3.9 NXtas

category `application` (application definition)

NXDL source: NXtas (http://svn.nexusformat.org/definitions/trunk/applications/NXtas.nxdl.xml)

version 1.0b

SVN Id Id: NXtas.nxdl.xml 487 2010-02-01 19:53:50Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXcrystal, NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample, NXsource

documentation This is an application definition for a triple axis spectrometer. It is for the trademark scan of the TAS, the Q-E scan. For your alignment scans use the rules in NXscan.

Table A.106: Tabular representation of NXtas[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
entry	NXentry		Look for special case table **entry[NXentry](within NXtas[definition])** below.

A.3.9.1 Special case table: entry[NXentry](within NXtas[definition])

Table A.107: Tabular representation of entry[NXentry](within NXtas[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
title	NX_CHAR		
start_time	NX_DATE_TIME		
definition	NXtas		Official NeXus DTD or NXDL schema to which this file conforms
	NXinstrument		Look for special case table **[NXinstrument](within entry[NXentry])** below.
	NXsample		Look for special case table **[NXsample](within entry[NXentry])** below.
	NXmonitor		Look for special case table **[NXmonitor](within entry[NXentry])** below.
	NXdata		One of the ei,ef,qh,qk,ql,en should get a primary=1 attribute to denote the main scan axis List of links: **ei** /NXentry/NXinstrument/NXcrystal:monochromator/ei **ef** /NXentry/NXinstrument/NXcrystal:analyzer/ef **en** /NXentry/NXsample/en **qh** /NXentry/NXsample/qh **qk** /NXentry/NXsample/qk **ql** /NXentry/NXsample/ql **data** /NXentry/NXinstrument/NXdetector/data

A.3.9.2 Special case table: [NXinstrument](within entry[NXentry])

Table A.108: Tabular representation of [NXinstrument](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
	NXsource		Look for special case table **[NXsource](within [NXinstrument])** below.
monochromator	NXcrystal		Look for special case table **monochromator[NXcrystal](within [NXinstrument])** below.
analyzer	NXcrystal		Look for special case table **analyzer[NXcrystal](within [NXinstrument])** below.

Table A.108: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
	NXdetector		Look for special case table **[NXdetector](within [NXinstrument])** below.

A.3.9.3 Special case table: [NXsource](within [NXinstrument])

Table A.109: Tabular representation of [NXsource](within [NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
`name`	NX_CHAR		
`probe`	neutron | x-ray		

A.3.9.4 Special case table: monochromator[NXcrystal](within [NXinstrument])

Table A.110: Tabular representation of monochromator[NXcrystal](within [NX-instrument])

Name and Attributes	Type	Units	Description (and Occurrences)
`ei`	NX_FLOAT	NX_ENERGY	Dimensions: size="1" • dim: index="1" value="np" value=""
`rotation_angle`	NX_FLOAT	NX_ANGLE	Dimensions: size="1" • dim: index="1" value="np" value=""

A.3.9.5 Special case table: analyzer[NXcrystal](within [NXinstrument])

Table A.111: Tabular representation of analyzer[NXcrystal](within [NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
`ef`	NX_FLOAT	NX_ENERGY	Dimensions: size="1" • dim: index="1" value="np" value=""
`rotation_angle`	NX_FLOAT	NX_ANGLE	Dimensions: size="1" • dim: index="1" value="np" value=""
`polar_angle`	NX_FLOAT	NX_ANGLE	Dimensions: size="1" • dim: index="1" value="np" value=""

A.3.9.6 Special case table: [NXdetector](within [NXinstrument])

Table A.112: Tabular representation of [NXdetector](within [NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
`data`	NX_INT		Dimensions: size="1" • dim: index="1" value="np" value=""
`polar_angle`	NX_FLOAT	NX_ANGLE	Dimensions: size="1" • dim: index="1" value="np" value=""

A.3.9.7 Special case table: [NXsample](within entry[NXentry])

Table A.113: Tabular representation of [NXsample](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
`name`	NX_CHAR		Descriptive name of sample
`qh`	NX_FLOAT	NX_DIMENSIONLESS	Dimensions: size="1" • dim: index="1" value="np" value=""
`qk`	NX_FLOAT	NX_DIMENSIONLESS	Dimensions: size="1" • dim: index="1" value="np" value=""
`ql`	NX_FLOAT	NX_DIMENSIONLESS	Dimensions: size="1" • dim: index="1" value="np" value=""
`en`	NX_FLOAT	NX_ENERGY	Dimensions: size="1" • dim: index="1" value="np" value=""
`rotation_angle`	NX_FLOAT	NX_ANGLE	Dimensions: size="1" • dim: index="1" value="np" value=""
`polar_angle`	NX_FLOAT	NX_ANGLE	Dimensions: size="1" • dim: index="1" value="np" value=""
`sgu`	NX_FLOAT	NX_ANGLE	Dimensions: size="1" • dim: index="1" value="np" value=""
`sgl`	NX_FLOAT	NX_ANGLE	Dimensions: size="1" • dim: index="1" value="np" value=""
`unit_cell`	NX_FLOAT	NX_LENGTH	Dimensions: size="1" • dim: index="1" value="6" value=""
`orientation_matrix`	NX_FLOAT	NX_DIMENSIONLESS	Dimensions: size="1" • dim: index="1" value="9" value=""

A.3.9.8 Special case table: [NXmonitor](within entry[NXentry])

Table A.114: Tabular representation of [NXmonitor](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
`mode`	monitor | timer		Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Table A.114: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
preset	NX_FLOAT		preset value for time or monitor
data	NX_FLOAT	NX_ANY	Total integral monitor counts Dimensions: size="1" • dim: index="1" value="np" value=""

A.3.10 NXtofraw

category application (application definition)

NXDL source: NXtofraw (http://svn.nexusformat.org/definitions/trunk/applications/NXtofraw.nxdl.xml)

version 1.0b

SVN Id Id: NXtofraw.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample, NXuser

documentation This is a application definition for raw data from a generic TOF instrument

Table A.115: Tabular representation of NXtofraw[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
entry	NXentry		Look for special case table **entry[NXentry](within NXtofraw[definition])** below.

A.3.10.1 Special case table: entry[NXentry](within NXtofraw[definition])

Table A.116: Tabular representation of entry[NXentry](within NXtofraw[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
title	NX_CHAR		
start_time	NX_DATE_TIME		
definition	NXtofraw		Official NeXus DTD or NXDL schema to which this file conforms
duration	NX_FLOAT		
run_number	NX_INT		
user	NXuser		Look for special case table **user[NXuser](within entry[NXentry])** below.
	NXinstrument		Look for special case table **[NXinstrument](within entry[NXentry])** below.
	NXsample		Look for special case table **[NXsample](within entry[NXentry])** below.
	NXmonitor		Look for special case table **[NXmonitor](within entry[NXentry])** below.

Table A.116: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
data	NXdata		List of links: **data** /NXentry/NXinstrument/NX-detector/data **detector_number** /NXentry/NXinstrument/NXd-etector/detector_number **time_of_flight** /NXentry/NXinstrument/NXd-etector/time_of_flight

A.3.10.2 Special case table: user[NXuser](within entry[NXentry])

Table A.117: Tabular representation of user[NXuser](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		

A.3.10.3 Special case table: [NXinstrument](within entry[NXentry])

Table A.118: Tabular representation of [NXinstrument](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
detector	NXdetector		Look for special case table **detector[NXdetector](within [NXinstrument])** below.

A.3.10.4 Special case table: detector[NXdetector](within [NXinstrument])

Table A.119: Tabular representation of detector[NXdetector](within [NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
data	NX_INT		Dimensions: size="2" • dim: index="1" value="ndet" value="" • dim: index="2" value="ntimechan" value=""
detector_number	NX_INT		Dimensions: size="1" • dim: index="1" value="ndet" value=""
distance	NX_FLOAT	NX_LENGTH	distance to sample for each detector Dimensions: size="1" • dim: index="1" value="ndet" value=""

Table A.119: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
time_of_flight	NX_FLOAT	NX_TIME_OF_FLIGHT	Dimensions: size="1" • dim: index="1" value="ntimechan" value=""
polar_angle	NX_FLOAT	NX_ANGLE	polar angle for each detector element Dimensions: size="1" • dim: index="1" value="ndet" value=""
azimuthal_angle	NX_FLOAT	NX_ANGLE	azimuthal angle for each detector element Dimensions: size="1" • dim: index="1" value="ndet" value=""
arrangement	NX_INT		This is new!! This array maps detector numbers to a position relative to a possible area geometry of the detector. A linear arrangement is covered too; then ny is 1 Dimensions: size="2" • dim: index="1" value="nx" value="" • dim: index="2" value="ny" value=""

A.3.10.5 Special case table: [NXsample](within entry[NXentry])

Table A.120: Tabular representation of [NXsample](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		Descriptive name of sample
nature	powder | liquid | "single crystal"		

A.3.10.6 Special case table: [NXmonitor](within entry[NXentry])

Table A.121: Tabular representation of [NXmonitor](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
mode	monitor | timer		Count to a preset value based on either clock time (timer) or received monitor counts (monitor).
preset	NX_FLOAT		preset value for time or monitor
distance	NX_FLOAT	NX_LENGTH	
data	NX_INT		Dimensions: size="1" • dim: index="1" value="ntimechan" value=""
time_of_flight	NX_FLOAT	NX_TIME_OF_FLIGHT	Dimensions: size="1" • dim: index="1" value="ntimechan" value=""

A.3.11 NXtomo

category `application` (application definition)

NXDL source: NXtomo (http://svn.nexusformat.org/definitions/trunk/applications/NXtomo.nxdl.xml)

version 1.0b

SVN Id

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample, NXsource

documentation This is the application definition for x-ray or neutron tomography raw data. In tomography first some dark field images are measured, some bright field images and, of course the sample. In order to properly sort the order of the images taken, a sequence number is stored with each image.

Table A.122: Tabular representation of NXtomo[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
`entry`	NXentry		Look for special case table **entry[NXentry](within NXtomo[definition])** below.

A.3.11.1 Special case table: entry[NXentry](within NXtomo[definition])

Table A.123: Tabular representation of entry[NXentry](within NXtomo[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
`title`	NX_CHAR		
`start_time`	NX_DATE_TIME		
`end_time`	NX_DATE_TIME		
`definition`	NXtomo		Official NeXus DTD or NXDL schema to which this file conforms
`instrument`	NXinstrument		Look for special case table **instrument[NXinstrument](within entry[NXentry])** below.
`sample`	NXsample		Look for special case table **sample[NXsample](within entry[NXentry])** below.
`control`	NXmonitor		Look for special case table **control[NXmonitor](within entry[NXentry])** below.
`data`	NXdata		List of links: **`data`** `/NXentry/NXinstrument/NXdetector:data/data` **`rotation_angle`** `/NXentry/NXsample/rotation_angle`

A.3.11.2 Special case table: instrument[NXinstrument](within entry[NXentry])

Table A.124: Tabular representation of instrument[NXinstrument](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
	NXsource		Look for special case table **[NXsource](within instrument[NXinstrument])** below.
bright_field	NXdetector		Look for special case table **bright_field[NXdetector](within instrument[NXinstrument])** below.
dark_field	NXdetector		Look for special case table **dark_field[NXdetector](within instrument[NXinstrument])** below.
sample	NXdetector		Look for special case table **sample[NXdetector](within instrument[NXinstrument])** below.

A.3.11.3 Special case table: [NXsource](within instrument[NXinstrument])

Table A.125: Tabular representation of [NXsource](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
type	NX_CHAR		
name	NX_CHAR		
probe	neutron | x-ray | electron		

A.3.11.4 Special case table: bright_field[NXdetector](within instrument[NXinstrument])

Table A.126: Tabular representation of bright_field[NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
data	NX_INT		Dimensions: size="3" • dim: index="1" value="nBrightFrames" value="" • dim: index="2" value="xsize" value="" • dim: index="3" value="ysize" value=""
sequence_number	NX_CHAR		In order to properly sort the order of the images taken in (for example) a tomography experiment, a sequence number is stored with each image. Dimensions: size="1" • dim: index="1" value="nBrightFrames" value=""

A.3.11.5 Special case table: dark_field[NXdetector](within instrument[NXinstrument])

Table A.127: Tabular representation of dark_field[NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
data	NX_INT		Dimensions: size="3" • dim: index="1" value="nDarkFrames" value="" • dim: index="2" value="xsize" value="" • dim: index="3" value="ysize" value=""
sequence_number	NX_CHAR		Dimensions: size="1" • dim: index="1" value="nDarkFrames" value=""

A.3.11.6 Special case table: sample[NXdetector](within instrument[NXinstrument])

Table A.128: Tabular representation of sample[NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
data	NX_INT		Dimensions: size="3" • dim: index="1" value="nSampleFrames" value="" • dim: index="2" value="xsize" value="" • dim: index="3" value="ysize" value=""
sequence_number	NX_CHAR		Dimensions: size="1" • dim: index="1" value="nSampleFrames" value=""
x_pixel_size	NX_FLOAT	NX_LENGTH	
y_pixel_size	NX_FLOAT	NX_LENGTH	
distance	NX_FLOAT	NX_LENGTH	Distance between detector and sample

A.3.11.7 Special case table: sample[NXsample](within entry[NXentry])

Table A.129: Tabular representation of sample[NXsample](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		Descriptive name of sample
rotation_angle	NX_FLOAT	NX_ANGLE	Dimensions: size="1" • dim: index="1" value="nSampleFrames" value=""

A.3.11.8 Special case table: control[NXmonitor](within entry[NXentry])

Table A.130: Tabular representation of control[NXmonitor](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
integral	NX_FLOAT	NX_ANY	Total integral monitor counts for each measured frame. This is in order to correct for fluctuations in the beam between frames Dimensions: size="1" • dim: index="1" value="nDarkFrames + nBrightFrames + nSampleFrame" value=""

A.3.12 NXtomophase

category application (application definition)

NXDL source: NXtomophase (http://svn.nexusformat.org/definitions/trunk/applications/NXtomophase.nxdl.xml)

version 1.0b

SVN Id

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample, NXsource

documentation This is the application definition for x-ray or neutron tomography raw data with phase contrast variation at each point. In tomography first some dark field images are measured, some bright field images and, of course the sample. In order to properly sort the order of the images taken, a sequence number is stored with each image.

Table A.131: Tabular representation of NXtomophase[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
entry	NXentry		Look for special case table **entry[NXentry](within NXtomophase[definition])** below.

A.3.12.1 Special case table: entry[NXentry](within NXtomophase[definition])

Table A.132: Tabular representation of entry[NXentry](within NXtomophase[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
title	NX_CHAR		
start_time	NX_DATE_TIME		
end_time	NX_DATE_TIME		
definition	NXtomophase		Official NeXus DTD or NXDL schema to which this file conforms
instrument	NXinstrument		Look for special case table **instrument[NXinstrument](within entry[NXentry])** below.
sample	NXsample		Look for special case table **sample[NXsample](within entry[NXentry])** below.

Table A.132: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
control	NXmonitor		Look for special case table **control[NXmonitor](within entry[NXentry])** below.
data	NXdata		List of links: **data** /NXentry/NXinstrument/NX-detector:sample/data **rotation_angle** /NXentry/NXsam-ple/rotation_angle

A.3.12.2 Special case table: instrument[NXinstrument](within entry[NXentry])

Table A.133: Tabular representation of instrument[NXinstrument](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
	NXsource		Look for special case table **[NXsource](within instrument[NXinstrument])** below.
bright_field	NXdetector		Look for special case table **bright_field[NXdetector](within instrument[NXinstrument])** below.
dark_field	NXdetector		Look for special case table **dark_field[NXdetector](within instrument[NXinstrument])** below.
sample	NXdetector		Look for special case table **sample[NXdetector](within instrument[NXinstrument])** below.

A.3.12.3 Special case table: [NXsource](within instrument[NXinstrument])

Table A.134: Tabular representation of [NXsource](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
type	NX_CHAR		
name	NX_CHAR		
probe	neutron | x-ray | electron		

A.3.12.4 Special case table: bright_field[NXdetector](within instrument[NXinstrument])

Table A.135: Tabular representation of bright_field[NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
data	NX_INT		Dimensions: size="3" • dim: index="1" value="nBrightFrames" value="" • dim: index="2" value="xsize" value="" • dim: index="3" value="ysize" value=""
sequence_number	NX_CHAR		Dimensions: size="1" • dim: index="1" value="nBrightFrames" value=""

A.3.12.5 Special case table: dark_field[NXdetector](within instrument[NXinstrument])

Table A.136: Tabular representation of dark_field[NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
data	NX_INT		Dimensions: size="3" • dim: index="1" value="nDarkFrames" value="" • dim: index="2" value="xsize" value="" • dim: index="3" value="ysize" value=""
sequence_number	NX_CHAR		Dimensions: size="1" • dim: index="1" value="nDarkFrames" value=""

A.3.12.6 Special case table: sample[NXdetector](within instrument[NXinstrument])

Table A.137: Tabular representation of sample[NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
data	NX_INT		Dimensions: size="4" • dim: index="1" value="nSampleFrames" value="" • dim: index="2" value="nPhase" value="" • dim: index="3" value="xsize" value="" • dim: index="4" value="ysize" value=""
sequence_number	NX_CHAR		Dimensions: size="2" • dim: index="1" value="nSampleFrames" value="" • dim: index="2" value="nPhase" value=""
x_pixel_size	NX_FLOAT	NX_LENGTH	
y_pixel_size	NX_FLOAT	NX_LENGTH	
distance	NX_FLOAT	NX_LENGTH	Distance between detector and sample

A.3.12.7 Special case table: sample[NXsample](within entry[NXentry])

Table A.138: Tabular representation of sample[NXsample](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		Descriptive name of sample
rotation_angle	NX_FLOAT	NX_ANGLE	Dimensions: size="1" • dim: index="1" value="nSampleFrames" value=""

A.3.12.8 Special case table: control[NXmonitor](within entry[NXentry])

Table A.139: Tabular representation of control[NXmonitor](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
integral	NX_FLOAT	NX_ANY	Total integral monitor counts for each measured frame. This is in order to correct for fluctuations in the beam between frames Dimensions: size="1" • dim: index="1" value="nDarkFrames + nBrightFrames + nSampleFrame" value=""

A.3.13 NXtomoproc

category application (application definition)

NXDL source: NXtomoproc (http://svn.nexusformat.org/definitions/trunk/applications/NXtomoproc.nxdl.xml)

version 1.0b

SVN Id

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXentry, NXinstrument, NXparameters, NXprocess, NXsample, NXsource

documentation This is an application definition for the final result of a tomography experiment: a 3D construction of some volume of physical properties.

Table A.140: Tabular representation of NXtomoproc[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
entry	NXentry		Look for special case table **entry[NXentry](within NXtomoproc[definition])** below.

A.3.13.1 Special case table: entry[NXentry](within NXtomoproc[definition])

Table A.141: Tabular representation of entry[NXentry](within NXtomoproc[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
title	NX_CHAR		
definition	NXtomoproc		Official NeXus DTD or NXDL schema to which this file conforms
	NXinstrument		Look for special case table **[NXinstrument](within entry[NXentry])** below.
	NXsample		Look for special case table **[NXsample](within entry[NXentry])** below.
reconstruction	NXprocess		Look for special case table **reconstruction[NXprocess](within entry[NXentry])** below.
data	NXdata		Look for special case table **data[NXdata](within entry[NXentry])** below.

A.3.13.2 Special case table: [NXinstrument](within entry[NXentry])

Table A.142: Tabular representation of [NXinstrument](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
	NXsource		Look for special case table **[NXsource](within [NXinstrument])** below.

A.3.13.3 Special case table: [NXsource](within [NXinstrument])

Table A.143: Tabular representation of [NXsource](within [NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
type	NX_CHAR		
name	NX_CHAR		
probe	neutron | x-ray | electron		

A.3.13.4 Special case table: [NXsample](within entry[NXentry])

Table A.144: Tabular representation of [NXsample](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		Descriptive name of sample

A.3.13.5 Special case table: reconstruction[NXprocess](within entry[NXentry])

Table A.145: Tabular representation of reconstruction[NXprocess](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
program	NX_CHAR		Name of the program used for reconstruction
version	NX_CHAR		Version of the program used
date	NX_DATE_TIME		Date and time of reconstruction processing.
parameters	NXparameters		Look for special case table **parameters[NXparameters](within reconstruction[NXprocess])** below.

A.3.13.6 Special case table: parameters[NXparameters](within reconstruction[NXprocess])

Table A.146: Tabular representation of parameters[NXparameters](within reconstruction[NXprocess])

Name and Attributes	Type	Units	Description (and Occurrences)
raw_file	NX_CHAR		Original raw data file this data was derived from

A.3.13.7 Special case table: data[NXdata](within entry[NXentry])

Table A.147: Tabular representation of data[NXdata](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
data	NX_INT		This is the reconstructed volume. This can be different things. Please indicates in the unit attribute what physical quantity this really is. Dimensions: size="3" • dim: index="1" value="nx" value="" • dim: index="2" value="nx" value="" • dim: index="3" value="nz" value=""
scaling_factor	NX_FLOAT		The elements in data are usually float values really. For efficiency reasons these are usually stored as integers after scaling with a scale factor. This value is this scale factor. It is required to get the actual physical value, when necessary.
offset	NX_FLOAT		An eventuell offset to apply to the data in data
x	NX_FLOAT	NX_ANY	This is an array holding the values to use for the x-axis of data. The units must be appropriate for the measurement. Dimensions: size="1" • dim: index="1" value="nx" value=""

Table A.147: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
y	NX_FLOAT	NX_ANY	This is an array holding the values to use for the y-axis of data. The units must be appropriate for the measurement. Dimensions: size="1" • dim: index="1" value="ny" value=""
z	NX_FLOAT	NX_ANY	This is an array holding the values to use for the z-axis of data. The units must be appropriate for the measurement. Dimensions: size="1" • dim: index="1" value="nz" value=""

A.3.14 NXxbase

category application (application definition)

NXDL source: NXxbase (http://svn.nexusformat.org/definitions/trunk/applications/NXxbase.nxdl.xml)

version 1.0b

SVN Id

NeXus Definition Language NXDL

extends class: NXobject

other classes included: NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXmonochromator, NXsample, NXsource

documentation This definition covers the common parts of all monochromatic single crystal raw data application definitions

Table A.148: Tabular representation of NXxbase[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
entry	NXentry		Look for special case table **entry[NXentry](within NXxbase[definition])** below.

A.3.14.1 Special case table: entry[NXentry](within NXxbase[definition])

Table A.149: Tabular representation of entry[NXentry](within NXxbase[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
title	NX_CHAR		
start_time	NX_DATE_TIME		
end_time	NX_DATE_TIME		
definition	NXxbase		Official NeXus DTD or NXDL schema to which this file conforms
instrument	NXinstrument		Look for special case table **instrument[NXinstrument](within entry[NXentry])** below.

Table A.149: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
sample	NXsample		Look for special case table **sample[NXsample](within entry[NXentry])** below.
control	NXmonitor		Look for special case table **control[NXmonitor](within entry[NXentry])** below.
	NXdata		The name of this group id data if there is only one detector; if there are several the names will be data1, data2, data3 and will data will point to the corresponding detector groups in the instrument hierarchy. List of links: **data** /NXentry/NXinstrument/NX-detector/data

A.3.14.2 Special case table: instrument[NXinstrument](within entry[NXentry])

Table A.150: Tabular representation of instrument[NXinstrument](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
source	NXsource		Look for special case table **source[NXsource](within instrument[NXinstrument])** below.
monochromator	NXmonochromator		Look for special case table **monochromator[NXmonochromator](within instrument[NXinstrument])** below.
detector	NXdetector		The name of the group is detector if there is only one detector, if there are several, names have to be detector1, detector2, ...detectorn Occurences: 1 : 99999999999999999999999 Look for special case table **detector[NXdetector](within instrument[NXinstrument])** below.

A.3.14.3 Special case table: source[NXsource](within instrument[NXinstrument])

Table A.151: Tabular representation of source[NXsource](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
type	NX_CHAR		
name	NX_CHAR		
probe	neutron | x-ray | electron		

A.3.14.4 Special case table: monochromator[NXmonochromator](within instrument[NXinstrument])

Table A.152: Tabular representation of monochromator[NXmonochromator](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
wavelength	NX_FLOAT	NX_WAVELENGTH	

A.3.14.5 Special case table: detector[NXdetector](within instrument[NXinstrument])

Table A.153: Tabular representation of detector[NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
data	NX_INT		The area detector data, the first dimension is always the number of scan points, the second and third are the number of pixels in x and y. The origin is always assumed to be in the center of the detector. maxOccurs is limited to the the number of detectors on your instrument Dimensions: size="3" • dim: index="1" value="np" value="" • dim: index="2" value="number of x pixels" value="" • dim: index="3" value="number of y pixels" value=""
@signal	NX_CHAR		
x_pixel_size	NX_FLOAT	NX_LENGTH	
y_pixel_size	NX_FLOAT	NX_LENGTH	
distance	NX_FLOAT	NX_LENGTH	

A.3.14.6 Special case table: sample[NXsample](within entry[NXentry])

Table A.154: Tabular representation of sample[NXsample](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
name	NX_CHAR		Descriptive name of sample
orientation_matrix	NX_FLOAT		The orientation matrix according to Busing and Levy conventions. This is not strictly necessary as the UB can always be derived from the data. But let us bow to common usage which includes thie UB nearly always. Dimensions: size="2" • dim: index="1" value="3" value="" • dim: index="2" value="3" value=""
unit_cell	NX_FLOAT		The unit cell, a, b ,b, alpha, beta, gamma. Again, not strictly necessary, but normally written. Dimensions: size="1" • dim: index="1" value="6" value=""

A.3.14.7 Special case table: control[NXmonitor](within entry[NXentry])

Table A.155: Tabular representation of control[NXmonitor](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
mode	monitor | timer		Count to a preset value based on either clock time (timer) or received monitor counts (monitor).
preset	NX_FLOAT		preset value for time or monitor
integral	NX_FLOAT	NX_ANY	Total integral monitor counts

A.3.15 NXxeuler

category application (application definition)

NXDL source: NXxeuler (http://svn.nexusformat.org/definitions/trunk/applications/NXxeuler.nxdl.xml)

version 1.0b

SVN Id

NeXus Definition Language NXDL

extends class: NXxbase

other classes included: NXdata, NXdetector, NXentry, NXinstrument, NXsample

documentation This is the application definition for raw data from a four circle diffractometer with an eulerian cradle. It extends NXxbase, so the full definition is the content of NXxbase plus the data defined here. All four angles are logged in order to support arbitray scans in reciprocal space.

Table A.156: Tabular representation of NXxeuler[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
entry	NXentry		Look for special case table **entry[NXentry](within NXxeuler[definition])** below.

A.3.15.1 Special case table: entry[NXentry](within NXxeuler[definition])

Table A.157: Tabular representation of entry[NXentry](within NXxeuler[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
definition	NXxeuler		Official NeXus DTD or NXDL schema to which this file conforms
instrument	NXinstrument		Look for special case table **instrument[NXinstrument](within entry[NXentry])** below.

Table A.157: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
sample	NXsample		Look for special case table **sample[NXsample](within entry[NXentry])** below.
name	NXdata		List of links: **polar_angle** /NXentry/NXinstrument/NXdetector/polar_angle **rotation_angle** /NXentry/NXsample/rotation_angle **chi** /NXentry/NXsample/chi **phi** /NXentry/NXsample/phi

A.3.15.2 Special case table: instrument[NXinstrument](within entry[NXentry])

Table A.158: Tabular representation of instrument[NXinstrument](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
detector	NXdetector		Look for special case table **detector[NXdetector](within instrument[NXinstrument])** below.

A.3.15.3 Special case table: detector[NXdetector](within instrument[NXinstrument])

Table A.159: Tabular representation of detector[NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
polar_angle	NX_FLOAT	NX_ANGLE	The polar_angle (two theta) where the detector is placed at each scan point. Dimensions: size="1" • dim: index="1" value="np" value=""

A.3.15.4 Special case table: sample[NXsample](within entry[NXentry])

Table A.160: Tabular representation of sample[NXsample](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
rotation_angle	NX_FLOAT	NX_ANGLE	This is an array holding the sample rotation angle at each scan point Dimensions: size="1" • dim: index="1" value="np" value=""

Table A.160: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
chi	NX_FLOAT	NX_ANGLE	This is an array holding the chi angle of the eulerian cradle at each scan point Dimensions: size="1" • dim: index="1" value="np" value=""
phi	NX_FLOAT	NX_ANGLE	This is an array holding the phi rotation of the eulerain cradle at each scan point Dimensions: size="1" • dim: index="1" value="np" value=""

A.3.16 NXxkappa

category `application` (application definition)

NXDL source: NXxkappa (http://svn.nexusformat.org/definitions/trunk/applications/NXxkappa.nxdl.xml)

version 1.0b

SVN Id

NeXus Definition Language NXDL

extends class: NXxbase

other classes included: NXdata, NXdetector, NXentry, NXinstrument, NXsample

documentation This is the application definition for raw data from a kappa geometry (CAD4) single crystal diffractometer. It extends NXxbase, so the full definition is the content of NXxbase plus the data defined here.

Table A.161: Tabular representation of NXxkappa[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
entry	NXentry		Look for special case table **entry[NXentry](within NXxkappa[definition])** below.

A.3.16.1 Special case table: entry[NXentry](within NXxkappa[definition])

Table A.162: Tabular representation of entry[NXentry](within NXxkappa[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
definition	NXxkappa		Official NeXus DTD or NXDL schema to which this file conforms
instrument	NXinstrument		Look for special case table **instrument[NXinstrument](within entry[NXentry])** below.
sample	NXsample		Look for special case table **sample[NXsample](within entry[NXentry])** below.

Table A.162: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
name	NXdata		List of links: **polar_angle** /NXentry/NXinstrument/NXdetector/polar_angle **rotation_angle** /NXentry/NXsample/rotation_angle **kappa** /NXentry/NXsample/kappa **phi** /NXentry/NXsample/phi

A.3.16.2 Special case table: instrument[NXinstrument](within entry[NXentry])

Table A.163: Tabular representation of instrument[NXinstrument](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
detector	NXdetector		Look for special case table **detector[NXdetector](within instrument[NXinstrument])** below.

A.3.16.3 Special case table: detector[NXdetector](within instrument[NXinstrument])

Table A.164: Tabular representation of detector[NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
polar_angle	NX_FLOAT	NX_ANGLE	The polar_angle (two theta) at each scan point Dimensions: size="1" • dim: index="1" value="np" value=""

A.3.16.4 Special case table: sample[NXsample](within entry[NXentry])

Table A.165: Tabular representation of sample[NXsample](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
rotation_angle	NX_FLOAT	NX_ANGLE	This is an array holding the sample rotation angle at each scan point Dimensions: size="1" • dim: index="1" value="np" value=""
kappa	NX_FLOAT	NX_ANGLE	This is an array holding the kappa angle at each scan point Dimensions: size="1" • dim: index="1" value="np" value=""

Table A.165: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
phi	NX_FLOAT	NX_ANGLE	This is an array holding the phi angle at each scan point Dimensions: size="1" • dim: index="1" value="np" value=""
alpha	NX_FLOAT	NX_ANGLE	This holds the inclination angle of the kappa arm.

A.3.17 NXxnb

category `application` (application definition)

NXDL source: NXxnb (http://svn.nexusformat.org/definitions/trunk/applications/NXxnb.nxdl.xml)

version 1.0b

SVN Id

NeXus Definition Language NXDL

extends class: NXxbase

other classes included: NXdata, NXdetector, NXentry, NXinstrument, NXsample

documentation This is the application definition for raw data from a single crystal diffractometer measuring in normal beam mode. It extends NXxbase, so the full definition is the content of NXxbase plus the data defined here. All angles are logged in order to support arbitray scans in reciprocal space.

Table A.166: Tabular representation of NXxnb[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
entry	NXentry		Look for special case table **entry[NXentry](within NXxnb[definition])** below.

A.3.17.1 Special case table: entry[NXentry](within NXxnb[definition])

Table A.167: Tabular representation of entry[NXentry](within NXxnb[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
definition	NXxnb		Official NeXus DTD or NXDL schema to which this file conforms
instrument	NXinstrument		Look for special case table **instrument[NXinstrument](within entry[NXentry])** below.
sample	NXsample		Look for special case table **sample[NXsample](within entry[NXentry])** below.

Table A.167: (continued)

Name and Attributes	Type	Units	Description (and Occurrences)
name	NXdata		List of links: **polar_angle** /NXentry/NXinstrument/NXdetector/polar_angle **tilt** /NXentry/NXinstrument/NXdetector/tilt **rotation_angle** /NXentry/NXsample/rotation_angle

A.3.17.2 Special case table: instrument[NXinstrument](within entry[NXentry])

Table A.168: Tabular representation of instrument[NXinstrument](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
detector	NXdetector		Look for special case table **detector[NXdetector](within instrument[NXinstrument])** below.

A.3.17.3 Special case table: detector[NXdetector](within instrument[NXinstrument])

Table A.169: Tabular representation of detector[NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
polar_angle	NX_FLOAT	NX_ANGLE	The polar_angle (gamma) of the detector for each scan point. Dimensions: size="1" • dim: index="1" value="np" value=""
tilt_angle	NX_FLOAT	NX_ANGLE	The angle by which the detector has been tilted out of the scattering plane. Dimensions: size="1" • dim: index="1" value="np" value=""

A.3.17.4 Special case table: sample[NXsample](within entry[NXentry])

Table A.170: Tabular representation of sample[NXsample](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
rotation_angle	NX_FLOAT	NX_ANGLE	This is an array holding the sample rotation angle at each scan point Dimensions: size="1" • dim: index="1" value="np" value=""

A.3.18 NXxrot

category `application` (application definition)

NXDL source: NXxrot (http://svn.nexusformat.org/definitions/trunk/applications/NXxrot.nxdl.xml)

version 1.0b

SVN Id

NeXus Definition Language NXDL

extends class: NXxbase

other classes included: NXdata, NXdetector, NXentry, NXinstrument, NXsample

documentation This is the application definition for raw data from a rotation camera. It extends NXxbase, so the full definition is the content of NXxbase plus the data defined here.

Table A.171: Tabular representation of NXxrot[definition]

Name and Attributes	Type	Units	Description (and Occurrences)
`entry`	NXentry		Look for special case table **entry[NXentry](within NXxrot[definition])** below.

A.3.18.1 Special case table: entry[NXentry](within NXxrot[definition])

Table A.172: Tabular representation of entry[NXentry](within NXxrot[definition])

Name and Attributes	Type	Units	Description (and Occurrences)
`definition`	NXxrot		Official NeXus DTD or NXDL schema to which this file conforms
`instrument`	NXinstrument		Look for special case table **instrument[NXinstrument](within entry[NXentry])** below.
`sample`	NXsample		Look for special case table **sample[NXsample](within entry[NXentry])** below.
`name`	NXdata		List of links: **`rotation_angle`** `/NXentry/NXsample/rotation_angle`

A.3.18.2 Special case table: instrument[NXinstrument](within entry[NXentry])

Table A.173: Tabular representation of instrument[NXinstrument](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
`detector`	NXdetector		Look for special case table **detector[NXdetector](within instrument[NXinstrument])** below.

A.3.18.3 Special case table: detector[NXdetector](within instrument[NXinstrument])

Table A.174: Tabular representation of detector[NXdetector](within instrument[NXinstrument])

Name and Attributes	Type	Units	Description (and Occurrences)
`polar_angle`	NX_FLOAT	NX_ANGLE	The polar_angle (two theta) where the detector is placed.

A.3.18.4 Special case table: sample[NXsample](within entry[NXentry])

Table A.175: Tabular representation of sample[NXsample](within entry[NXentry])

Name and Attributes	Type	Units	Description (and Occurrences)
`rotation_angle`	NX_FLOAT	NX_ANGLE	This is an array holding the sample rotation angle at each scan point Dimensions: size="1" • dim: index="1" value="np" value=""

A.4 NeXus Contributed Classes

A description of each NeXus Contributed Class is given.

No NeXus Contributed Classes were defined at the time this version of the manual was created.

Appendix B

NeXus Utilities

Most of these utility programs are run from the command line. It will be noted if a program provides a graphical user interface (GUI).

nx2dtd Utility to convert a NeXus file into XML

nxbrowse NeXus Browser

nxconvert Utility to convert a NeXus file into HDF4/HDF5/XML/...

nxdir `NXdir` is a utility for querying a NeXus file about its contents. Full documentation can be found by running this command:

```
nxdir -h
```

nxingest `nxingest` extracts the metadata from a NeXus file to create an XML file according to a mapping file.

The mapping file defines the structure (names and hierarchy) and content (from either the NeXus file, the mapping file or the current time) of the output file. See the man page for a description of the mapping file.

This tool uses the NAPI. Thus, any of the supported formats (HDF4, HDF5 and XML) can be read.

nxsummary Use `nxsummary` to generate summary of a NeXus file.

This program relies heavily on a configuration file. Each `item` tag in the file describes a node to print from the NeXus file. The `path` attribute describes where in the NeXus file to get information from. The `label` attributes is what will be printed when showing the value of the specified field. The optional `operation` attribute provides for certain operations to be performed on the data before printing out the result.

See the source code documentation for more details.

nxtranslate `nxtranslate` is an anything to NeXus converter. This is accomplished by using translation files and a plugin style of architecture where `nxtranslate` can read from new formats as plugins become available. The documentation for `nxtranslate` describes its usage by three types of individuals:

the person using existing translation files to create NeXus files
the person creating translation files
the person writing new `Retrievers`

All of these concepts are discussed in detail in the documentation provided with the source code.

nxvalidate From the source code documentation: 'Utility to convert a NeXus file into HDF4/HDF5/XML/...' Note there is also a newer Java program called `NXvalidate`.

NXdump `NXdump` is temporary wrapper script for `.libs/NXdump` in the NeXus code repository. From the source code documentation: ' The NXdump program cannot be directly executed until all the libtool libraries that it depends on are installed. This wrapper script should never be moved out of the build directory. If it is, it will not operate correctly. '

NXplot An extendable utility for plotting any NeXus file. `NXplot` is an Eclipse-based GUI project in Java to plot data in NeXus files. (The project was started at the first NeXus Code Camp in 2009.)

Appendix C

NeXus: the basics for the truly impatient

Sometimes, people are just too impatient to wade through a big manual. This chapter is your salvation. Section C.1 describes the basic organization within the NeXus hierarchy. Section C.2 describes the NeXus coordinate system. Section C.3 describes the different purposes of NeXus base classes and application definitions. Section C.4 works through an example to construct a NXDL file for a fictional scientific instrument that closely resembles a neutron powder diffractometer. Section C.5 gives advice about storing the results from data processing.

This section was derived, almost verbatim, from the excellent first tutorial, `NXDLTutorial.pdf`[1] on how to write a definition file in NXDL (in the NeXus *definitions* repository).

C.1 Basic organization within the NeXus hierarchy

Let us start with a recapitulation of some of the NeXus features relevant to data files for analysis. NeXus experts can skip this section. The first are some of the NeXus guiding principles.

A NeXus file has to contain all the data necessary for standard data analysis.
A NeXus file is extendable.

Data in NeXus files are stored in a structured form. To this purpose, NeXus uses the concept of groups. A NeXus *group* is a container which can contain other groups or data sets (called *field*). NeXus identifies groups through two attributes: `name` and class (called `type` in the NeXus specification for groups). `NXentry` is an example of a class while `entry` is an example of a name.

NeXus *group* names and *field* names must conform to the regular expression syntax of Example C.1.

Example C.1 Regular expression pattern for NXDL group and field names

```
[A-Za-z_][\w_]*
```

Note that this name pattern starts with a letter (upper or lower case) or "_", then letters, numbers, and "_" and is limited to a limit of 63 characters imposed by the HDF5 rules for names.

The naming rule is different for classes: class names are part of the NeXus standard and always start with the prefix NX. At first glance this scheme seems odd. After all, a data analysis program becomes much easier to write if all the names would be known in advance. However, there is a good reason for the class/name scheme and this is the fact that multiple elements of the same type may occur. A reflectometer has many slits of type `NXaperture`. Lots of instruments have multiple detector banks of type `NXdetector`. A data analysis program has to figure that out and resolve which is the data to be evaluated.

The NeXus suggestion is to pick the name from the class unless there is some reason to choose otherwise. For example, `NXentry` would have a name of `entry`. HDF imposes a rule that says no two entities at the same level of an HDF file can have the

[1] https://svn.nexusformat.org/definitions/trunk/tutorial/NXDLTutorial.pdf

same name. The NeXus suggestion for a default name is to follow the name with an index number starting from 1. For example, with two `NXentry` groups, by default they would be named `name2` and `name2`. Please remember that this is only a suggestion, not a requirement.

A NeXus file has the following structure:

- `NXentry`
  - `NXuser`
  - `NXmonitor`
  - `NXsample`
  - `NXdata`
  - `NXinstrument`
    - `NX`*component*
    - `NX`*component*

This shows that at the root level of a NeXus file is a group of type `NXentry`. In fact, one or more such groups are allowed, where each usually represents a separate *measurement*.[2] These are NeXus structures for storing multiple (possibly related) data sets in one file. Note that a NeXus file must contain at least one `NXentry` group. The `NXentry` group will always be at the root level of a NeXus HDF file or will always be a child of the `NXroot` root element of a NeXus XML data file.

The `NXentry` group contains further groups:

- one or more `NXuser` groups containing information (metadata) about the experimenter
- An `NXsample` group which contains metadata about the sample
- One or more `NXmonitor` groups which contains data about the counting statistics and how counting happened.
- An `NXinstrument` which contains further groups, one for each relevant instrument component.
- One or more `NXdata` groups that describe the location (using *links*) of the default plottable data within this `NXentry`.[3] The information provided by the `NXdata` group, identification of the default plottable data, is one of the basic motivations (see Section 1.1.1) for the NeXus standard.

C.2 NeXus coordinates

NeXus supports two coordinate systems. If you need to be very exact, use the *McStas* coordinate system. *McStas* is mostly of interest to simulators of neutron instruments wishing to store the results of simulations in NeXus files. *McStas* provides no value to X-ray users. Most people will use the simple coordinate system as shown in Figure C.1.

[2] A strict definition of *measurement* is not provided by NeXus. Generally, a measurement is a single dataset and all related metadata.

[3] The choice of the name of the `NXdata` class is historic so renaming it to something more descriptive of its actual function would break legacy data files.

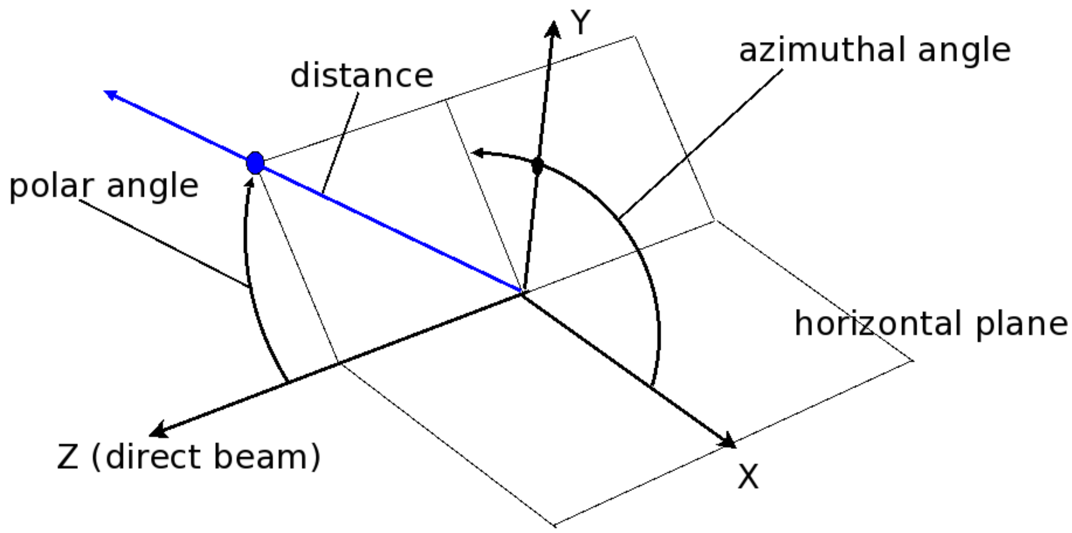

Figure C.1: NeXus simple coordinate system

This appears complicated but becomes very simple if the instrument does not move out of the scattering plane: Then the NeXus polar angle becomes what is commonly known as *two theta*. There is another gotcha: NeXus stores the polar angle downstream: for example if you have a monochromator pointing at a sample, the polar angle of the monochromator is stored at the sample. The NIAC decided upon this after lengthy discussions because it is the only way how to cope with instruments which have multiple backends, like multiple analyzers. The zero point for distances is the sample. Distances are in relation to this zero point. Negative distances point towards the source, positive distances behind the sample.

C.3 Note about NXDL Classes

NeXus base class and application definitions are written in NeXus Definition Language, NXDL. NXDL is in fact an application of XML to the problem of writing application definitions. The nice thing about NXDL is that it can be converted to an XML schema which then can be used to validate NeXus file against the definition.

base classes are dictionaries of names to use for the various fields in a NeXus group. Consequently there is a base class for each defined NeXus groups. Base classes define names for anything which can possibly be used to describe this component. Thus base classes tend to be pretty big. Do not worry, we reduce this later.

application definition is a definition of the content of a NeXus file as used for a special instrument type or an exchange format for data later in the data analysis pipeline. This content is what a NeXus file producer has to provide in order to write a valid NeXus file for this type of application. In turn a data analysis software author can rely on this information to be present in a valid NeXus file for this type of application. Another use of an application definition is to use define the data interface between a source of data and a consumer.

C.4 Creating a NXDL Specification

One easy way to describe how to store data in the NeXus class structure and to create a NXDL specification is to work through an example. Along the way, we will describe some key decisions that influence our particular choices of metadata selection and data organization. So, on with the example ...

Consider yourself to be in a position at the HYpothetical NEw Source (HYNES). You are tasked to write an application definition for raw data from the WOnderful New Instrument (WONI). This being a tutorial, WONI is actually a simple powder diffractometer but for reasons that only justify creating a new NXDL in this example. WONI cannot use any of the existing NXDL application definitions so we will make a new one. Refer to the WONI schematic in Figure C.2.

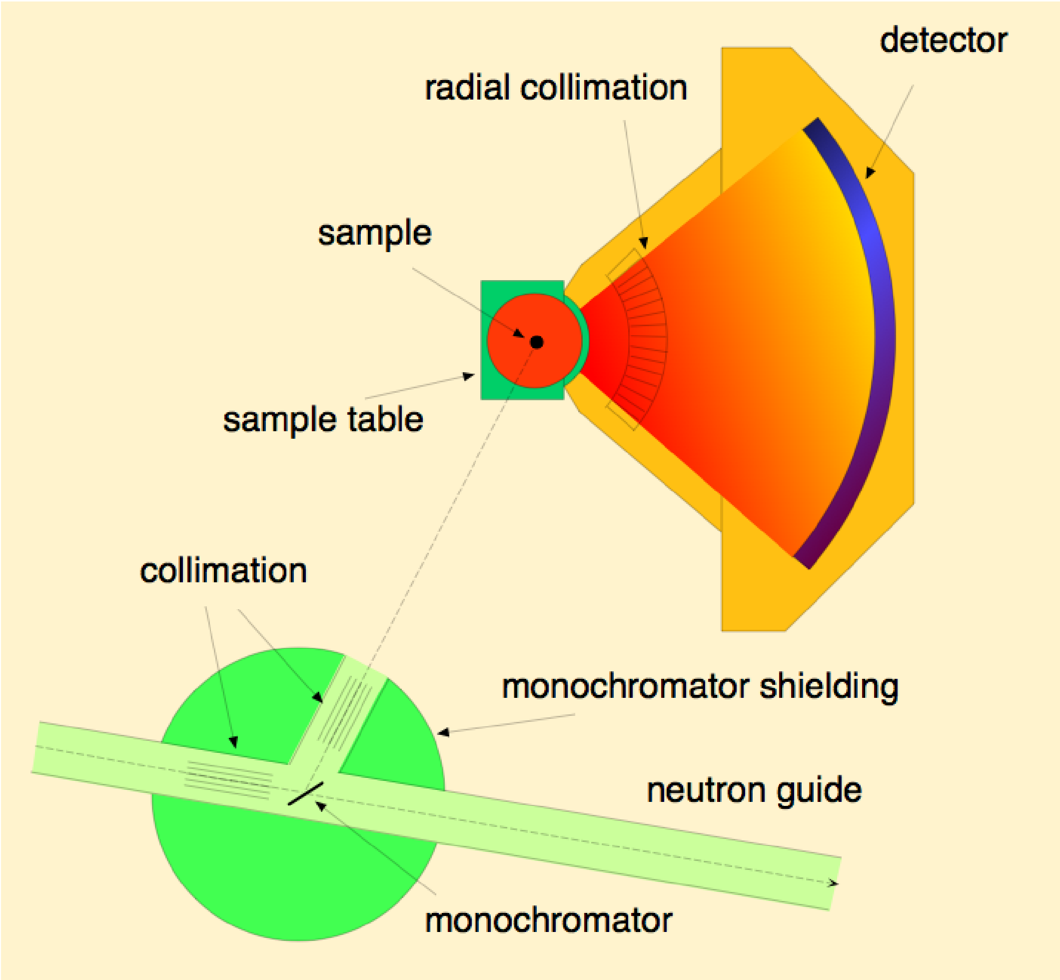

Figure C.2: The (fictional) WONI example powder diffractometer at HYNES

So there is a monochromator which generates a monochromatic beam which hits the sample which diffracts the beam. The diffracted beam is collected in a large banana-shaped position sensitive detector. Typical data looks like Figure C.3. There is a generous background to the data plus quite a number of diffraction peaks.

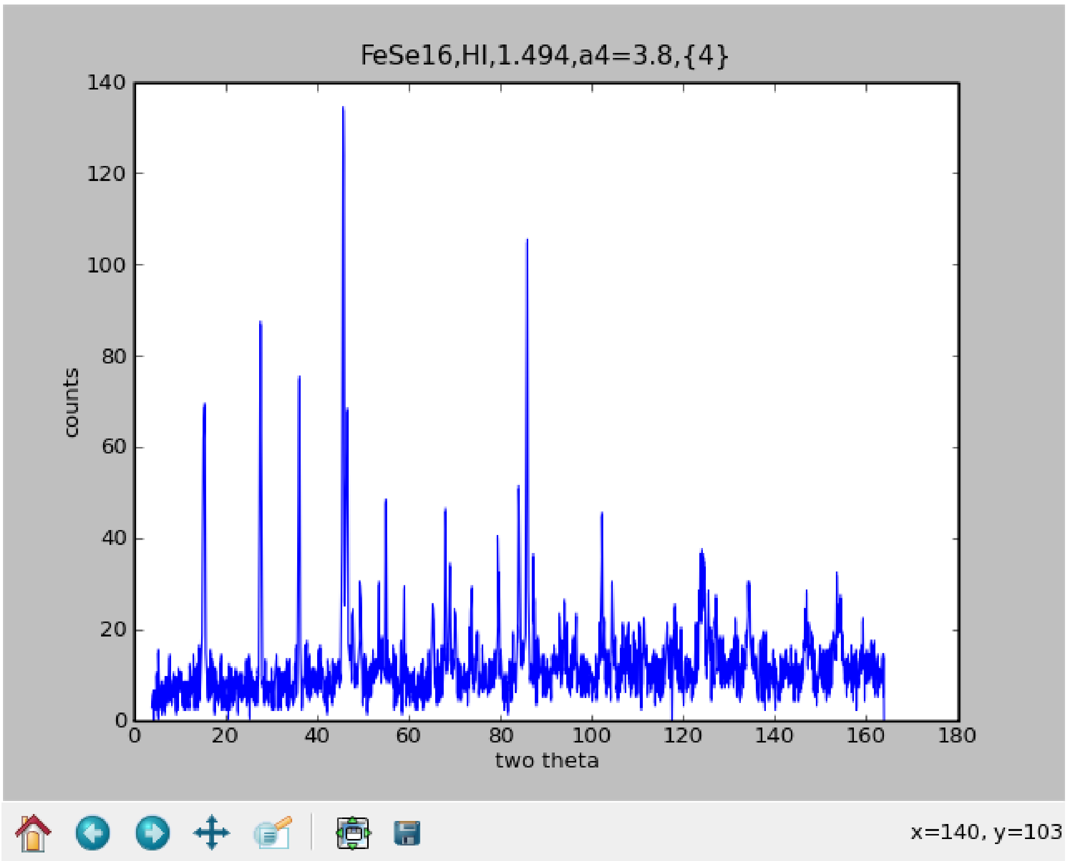

Figure C.3: Example Powder Diffraction Plot from (fictional) WONI at HYNES

C.4.1 Application Definition Steps

With all this introductory stuff out of the way, let us look at the process required to define an application definition:

1. *Think!* hard about what has to go into the data file.
2. *Map* the required fields into the NeXus hierarchy
3. *Describe* this map in a NXDL file
4. *Standardize* your definition through communication with the NIAC

C.4.2 Step 1: *Think!* hard about data

This is actually the hard bit. There are two things to consider:

1. What has to go into the data file?
2. What is the normal plot for this type of data?

For the first part, one of the NeXus guiding principles gives us - Guidance! 'A NeXus file must contain all the data necessary for standard data analysis.'

Not more and not less for an application definition. Of course the definition of *standard* data for analysis or a *standard* plot depends on the science and the type of data being described. Consult senior scientists in the field about this is if you are unsure.

Perhaps you must call an international meeting with domain experts to haggle that out. When considering this, people tend to put in everything which might come up. This is not the way to go.

A key test question is: Is this data item necessary for common data analysis? Only these necessary data items belong in an application definition.

The purpose of an application definition is that an author of upstream software who consumes the file can expect certain data items to be there at well defined places. On the other hand if there is a development in your field which analyzes data in a novel way and requires more data to do it, then it is better to err towards the side of more data.

Now for the case of WONI, the standard data analysis is either Rietveld refinement or profile analysis. For both purposes, the kind of radiation used to probe the sample (for WONI, neutrons), the wavelength of the radiation, the monitor (which tells us how long we counted) used to normalize the data, the counts and the two theta angle of each detector element are all required. Usually, it is desirable to know what is being analyzed, so some metadata would be nice: a title, the sample name and the sample temperature. The data typically being plotted is two theta against counts, as shown in Figure C.3 above. Summarizing, the basic information required from WONI is given in Table C.1.

Table C.1: basic information required from WONI

`title` of measurement
sample `name`
sample `temperature`
`monitor`
type of radiation `probe`
`wavelength` of radiation incident on sample
`two theta` of detector elements
`counts` for each detector element

If you start to worry that this is too little information, hold on, the section on Using an Application Definition (Section C.4.7) will reveal the secret how to go from an application definition to a practical file.

C.4.3 Step 2: *Map* Data into the NeXus Hierarchy

This step is actually easier then the first one. We need to map the data items which were collected in Step 1 into the NeXus hierarchy. A NeXus file hierarchy starts with an `NXentry` group. At this stage it is advisable to pull up the base class definition for `NXentry` and study it. The first thing you might notice is that `NXentry` contains a field named `title`. Reading the documentation, you quickly realize that this is a good place to store our title. So the first mapping has been found.

```
title = /NXentry/title
```

Note
In this example, the mapping descriptions just contain the path strings into the NeXus file hierarchy with the class names of the groups to use. As it turns out, this is the syntax used in NXDL link specifications. How convenient!

Another thing to notice in the `NXentry` base class is the existence of a group of class `NXsample`. This looks like a great place to store information about the sample. Studying the `NXsample` base class confirms this view and there are two new mappings:

```
sample name = /NXentry/NXsample/name
sample temperature = /NXentry/NXsample/temperature
```

Scanning the `NXentry` base class further reveals there can be a `NXmonitor` group at this level. Looking up the base class for `NXmonitor` reveals that this is the place to store our monitor information.

```
monitor = /NXentry/NXmonitor/data
```

For the other data items, there seem to be no solutions in `NXentry`. But each of these data items describe the instrument in more detail. NeXus stores instrument descriptions in the `/NXentry/NXinstrument` branch of the hierarchy. Thus, we continue by looking at the definition of the `NXinstrument` base class. In there we find further groups for all possible instrument components. Looking at the schematic of WONI (Figure C.2), we realize that there is a source, a monochromator and a detector. Suitable groups can be found for these components in `NXinstrument` and further inspection of the appropriate base classes reveals the following further mappings:

```
probe = /NXentry/NXinstrument/NXsource/probe
wavelength = /NXentry/NXinstrument/NXcrystal/wavelength
two theta of detector elements =
/NXentry/NXinstrument/NXdetector/polar angle
counts for each detector element = /NXentry/NXinstrument/NXdetector/data
```

Thus we mapped all our data items into the NeXus hierarchy! What still needs to be done is to decide upon the content of the `NXdata` group in `NXentry`. This group describes the data necessary to make a quick plot of the data. For WONI this is `counts` versus `two theta`. Thus we add this mapping:

```
two theta of detector elements = /NXentry/NXdata/polar angle
counts for each detector element = /NXentry/NXdata/data
```

The full mapping of WONI data into NeXus is documented in Table C.2.

Table C.2: Full mapping of WONI data into NeXus

WONI data	NeXus path
`title` of measurement	`/NXentry/title`
sample `name`	`/NXentry/NXsample/name`
sample `temperature`	`/NXentry/NXsample/temperature`
`monitor`	`/NXentry/NXmonitor/data`
type of radiation `probe`	`/NXentry/MXinstrument/NXsource/probe`
`wavelength` of radiation incident on sample	`/NXentry/MXinstrument/NXcrystal/wavelength`
`two theta` of detector elements	`/NXentry/NXinstrument/NXdetector/polar_angle`
`counts` for each detector element	`/NXentry/NXinstrument/NXdetector/data`
`two theta` of detector elements	`/NXentry/NXdata/polar_angle`
`counts` for each detector element	`/NXentry/NXdata/data`

Looking at this one might get concerned that the two theta and counts data is stored in two places and thus duplicated. Stop worrying, this problem is solved at the NeXus API level. Typically `NXdata` will only hold links to the corresponding data items in `/NXentry/NXinstrument/NXdetector`.

In this step problems might occur. The first is that the base class definitions contain a bewildering number of parameters. This is on purpose: the base classes serve as dictionaries which define names for everything which possibly can occur. You do not have to give all that information. The key question is, as already said, What is required for typical data analysis for this type of application? You might also be unsure how to correctly store a particular data item. In such a case, contact the NIAC for help. Another problem which can occur is that you require to store information for which there is no name in one of the existing base classes or you have a new instrument component for which there is no base class alltogether. In such a case, please feel free to contact the NIAC with a suggestion for an extension of the base classes in question.

C.4.4 Step 3: *Describe* this map in a NXDL file

This is even easier. Some XML editing is necessary. Fire up your XML editor of choice and open a file. If your XML editor supports XML schema while editing XML, it is worth to load `nxdl.xsd`. Now your XML editor can help you to create a proper NXDL file. As always, the start is an empty template file. This looks like Example C.2. This is just the basic XML for a NXDL definition. It is advisable to change some of the documentation strings.

Example C.2 NXDL template file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!--
# NeXus - Neutron and X-ray Common Data Format
#
# Copyright (C) 2008-10 NeXus International Advisory Committee (NIAC)
#
# This library is free software; you can redistribute it and/or
# modify it under the terms of the GNU Lesser General Public
# License as published by the Free Software Foundation; either
# version 2 of the License, or (at your option) any later version.
#
# This library is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
# Lesser General Public License for more details.
#
# You should have received a copy of the GNU Lesser General Public
# License along with this library; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
#
# For further information, see http://www.nexusformat.org

########### SVN repository information ###################
# $Date: 2010-02-03 03:14:22 -0600 (Wed, 03 Feb 2010) $
# $Author: Pete Jemian $
# $Revision: 491 $
# $HeadURL: https://svn.nexusformat.org/definitions/trunk/manual/examples/NX__template__.nxdl.xml $
# $Id: NX__template__.nxdl.xml 491 2010-02-03 09:14:22Z Pete Jemian $
########### SVN repository information ###################
-->
<definition name="NX__template__" extends="NXobject" type="group"
    category="application"
    xmlns="http://definition.nexusformat.org/nxdl/3.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://definition.nexusformat.org/nxdl/3.1 ../nxdl.xsd"
    version="1.0b"
    >
    <doc>template for a NXDL application definition</doc>
</definition>
```

For example, copy and rename the file to `NXwoni.nxdl.xml`. Then, locate the XML root element `definition` and change the `name attribute` (the XML shorthand for this attribute is `/definition/@name`) to `NXwoni`. Change the `doc` as well. Also consider keeping track of `/definition/@version` as suits your development of this NXDL file.

The next thing which needs to be done is adding groups into the definition. A group is defined by some XML like this:

```xml
<group type="NXdata">

</group>
```

The type is the actual NeXus base class this group belongs to. Optionally a `name` attribute may be given (default is `data`).

Next, one needs to include data items too. The XML for such a data item looks like this:

```
<field name="polar_angle" type="NX_FLOAT units="NX_ANGLE">
  <doc>Link to polar angle in /NXentry/NXinstrument/NXdetector</doc>
  <dimensions size="1">
    <dim index="1" value="ndet"/>
  </dimensions>
</field>
```

The meaning of the `name` attribute is intuitive, the `type` can be looked up in the relevant base class definition. A `field` definition can optionally contain a `doc` element which contains a description of the data item. The `dimensions` entry specifies the dimensions of the data set. The `size` attribute in the dimensions tag sets the rank of the data, in this example: `1`. In the dimensions group there must be rank dim fields. Each `dim` tag holds two attributes: `index` determines to which dimension this tag belongs, the `1` means the first dimension. The `value` attribute then describes the size of the dimension. These can be plain integers, variables, such as in the example `ndet` or even expressions like `tof+1`.

Thus a NXDL file can be constructed. The full NXDL file for the WONI example is given in Section C.4.6. Clever readers may have noticed the strong similarity between our working example `NXwoni` and `NXmonopd` since they are essentially identical. Give yourselves a cookie if you spotted this.

C.4.5 Step 4: *Standardize* with the NIAC

Basically you are done. Your first application definition for NeXus is constructed. In order to make your work a standard for that particular application type, some more steps are required:

- Send your application definition to the NIAC for review
- Correct your definition per the comments of the NIAC
- Cure and use the definition for a year
- After a final review, it becomes the standard

The NIAC must review an application definition before it is accepted as a standard. The one year curation period is in place in order to gain practical experience with the definition and to sort out bugs from Step 1. In this period, data shall be written and analyzed using the new application definition.

C.4.6 Full listing of the WONI Application Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
# NeXus - Neutron and X-ray Common Data Format
#
# Copyright (C) 2008 NeXus International Advisory Committee (NIAC)
#
# This library is free software; you can redistribute it and/or
# modify it under the terms of the GNU Lesser General Public
# License as published by the Free Software Foundation; either
# version 2 of the License, or (at your option) any later version.
#
# This library is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
# Lesser General Public License for more details.
#
# You should have received a copy of the GNU Lesser General Public
# License along with this library; if not, write to the Free Software
```

```
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
#
# For further information, see http://www.nexusformat.org

########### SVN repository information ###################
# $Date: 2010-01-13 03:19:56 -0600 (Wed, 13 Jan 2010) $
# $Author: Pete Jemian $
# $Revision: 458 $
# $HeadURL: https://svn.nexusformat.org/definitions/trunk/applications/NXmonopd.nxdl.xml $
# $Id: NXmonopd.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian $
########### SVN repository information ###################
-->
<definition name="NXmonopd" restricts="NXentry" type="group"
    category="application"
    xmlns="http://definition.nexusformat.org/nxdl/3.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://definition.nexusformat.org/nxdl/3.1 ../nxdl.xsd"
    version="1.0b"
    svnid="$Id: NXmonopd.nxdl.xml 458 2010-01-13 09:19:56Z Pete Jemian $">
    <doc> Monochromatic Neutron and X-Ray Powder Diffraction. Instrument definition for a powder
        diffractometer at a monochromatic neutron or X-ray beam. This is both suited for a powder
        diffractometer with a single detector or a powder diffractometer with a position sensitive
        detector. </doc>
        <field name="title"/>
        <field name="start_time" type="NX_DATE_TIME"/>
        <field name="definition">
            <doc>Official NeXus DTD or NXDL schema to which this file conforms</doc>
            <enumeration>
                <item value="NXmonopd"></item>
            </enumeration>
        </field>
        <group type="NXinstrument">
            <group type="NXsource">
                <field name="type"/>
                <field name="name"/>
                <field name="probe">
                    <enumeration>
                        <item value="neutron"/>
                        <item value="x-ray"/>
                        <item value="electron"/>
                    </enumeration>
                </field>
            </group>
            <group type="NXcrystal">
                <field name="wavelength" type="NX_FLOAT" units="NX_WAVELENGTH">
                    <doc>Optimum diffracted wavelength</doc>
                    <dimensions size="1">
                        <dim index="1" value="i"/>
                    </dimensions>
                </field>
            </group>
            <group type="NXdetector">
                <field name="polar_angle" type="NX_FLOAT" axis="1">
                  <doc>where ndet = number of detectors</doc>
                  <dimensions size="1">
                    <dim index="1" value="ndet" />
                  </dimensions>
                </field>
                <field name="data" type="NX_INT" signal="1">
```

```
                  <doc>
                    detector signal (usually counts) are already
                    corrected for detector efficiency
                  </doc>
                  <dimensions size="1">
                    <dim index="1" value="ndet" />
                  </dimensions>
                </field>
            </group>
        </group>
        <group type="NXsample">
            <field name="name">
                <doc>Descriptive name of sample</doc>
            </field>
            <field name="rotation_angle" type="NX_FLOAT" units="NX_ANGLE">
                <doc> Optional rotation angle for the case when the powder diagram has been obtained
                    through an omega-2theta scan like from a traditional single detector  powder
                    diffractometer </doc>
            </field>
        </group>
        <group type="NXmonitor">
            <field name="mode">
                <doc>Count to a preset value based on either clock time (timer) or received monitor
                    counts (monitor).</doc>
                <enumeration>
                    <item value="monitor"/>
                    <item value="timer"/>
                </enumeration>
            </field>
            <field name="preset" type="NX_FLOAT">
                <doc>preset value for time or monitor</doc>
            </field>
            <field name="integral" type="NX_FLOAT" units="NX_ANY">
                <doc>Total integral monitor counts</doc>
            </field>
        </group>
        <group type="NXdata">
            <link name="polar_angle" target="/NXentry/NXinstrument/NXdetector/polar_angle">
                <doc>Link to polar angle in /NXentry/NXinstrument/NXdetector</doc>
            </link>
            <link name="data" target="/NXentry/NXinstrument/NXdetector/data">
                <doc>Link to data in /NXentry/NXinstrument/NXdetector</doc>
            </link>
        </group>
</definition>
```

C.4.7 Using an Application Definition

The application definition is like an interface for your data file. In practice files will contain far more information. For this, the extendable capability of NeXus comes in handy. More data can be added, and upstream software relying on the interface defined by the application definition can still retrieve the necessary information without any changes to their code.

NeXus application definitions only standardize classes. You are free to decide upon names of groups, subject to them matching regular expression for NeXus name attributes (Example C.1). Note the length limit of 63 characters imposed by HDF5. Please use sensible, descriptive names and separate multi worded names with underscores.

Something most people wish to add is more metadata, for example in order to index files into a database of some sort. Go ahead, do so, if applicable, scan the NeXus base classes for standardized names. For metadata, consider to use the `NXarchive` definition. In this context, it is worth to mention that a practical NeXus file might adhere to more then one application definition. For example, WONI data files may adhere to both the `NXmonopd` and `NXarchive` definitions. The first for data analysis, the second for indexing into the database.

Often, instrument scientists want to store the complete state of their instrument in data files in order to be able to find out what went wrong if the data is unsatisfactory. Go ahead, do so, please use names from the NeXus base classes.

Site policy might require you to store the names of all your bosses up to the current head of state in data files. Go ahead, add as many `NXuser` classes as required to store that information. Knock yourselves silly over this.

Your Scientific Accounting Department (SAD) may ask of you the preposterous; to store billing information into data files. Go ahead, do so if your judgment allows. Just do not expect the NIAC to provide base classes for this and do not use the prefix NX for your classes.

In most cases, NeXus files will just have one `NXentry` class group. But it may be required to store multiple related data sets of the results of data analysis into the same data file. In this case create more entries. Each entry should be interpretable standalone, i.e. contain all the information of a complete `NXentry` class. Please keep in mind that groups or data items which stay constant across entries can always be linked in.

C.5 Processed Data

Data reduction and analysis programs are encouraged to store their results in NeXus data files. As far as the necessary, the normal NeXus hierarchy is to be implemented. In addition, processed data files must contain a `NXprocess` group. This group, that documents and preserves data provenance, contains the name of the data processing program and the parameters used to run this program in order to achieve the results stored in this entry. Multiple processing steps must have a separate entry each.

Appendix D

Frequently Asked Questions

This is a list of commonly asked questions concerning the NeXus data format.

1. *How many facilities use NeXus?*

   This is continually evolving. It has been used as the instrument format for several years on some or all instruments at a number of facilities including PSI (Switzerland), LLB (France), LANSCE (USA), and APS (USA). It will be used on all future instrumentation at ISIS (UK), NIST (USA), and ANSTO (Australia). Finally, it has been formally adopted by major facilities under construction, the SNS (USA), JPARC (Japan) and Diamond Light Source (UK).

2. *NeXus files are only useful for archiving instrumental data, aren't they?*

   NeXus files can be used to store both extremely simple data, e.g. a single (x,y) array, and highly complex instrument descriptions. In fact, the original intention of the NeXus data format was to provide a way of interchanging data between facilities and their user communities. However, the power of NeXus hierarchical design has led to its adoption as a standard archiving format by several major facilities, such as ISIS, LANSCE, and the SNS.

3. *Why aren't `NXsample` and `NXmonitor` groups stored in the `NXinstrument` group?*

   A NeXus file can contain a number of `NXentry` groups, which may represent different scans in an experiment, or sample and calibration runs, etc. In many cases, though by no means all, the instrument has the same configuration so that it would be possible to save space by storing the `NXinstrument` group once and using multiple links in the remaining `NXentry` groups. It is assumed that the sample and monitor information would be more likely to change from run to run, and so should be stored at the top level.

4. *How do I identify the plottable data?*

   Any program whose aim is to identify plottable data should use the following procedure:

   (a) Open the first top level NeXus group with class `NXentry`.
   (b) Open the first NeXus group with class `NXdata`.
   (c) Loop through NeXus fields in this group searching for the item with attribute `signal="1"` indicating this field has the plottable data.
   (d) Check to see if this field has an attribute called `axes`. If so, the names are defined as a comma-delimited string within this attribute in the C-order of the data array, and you can skip the next two steps.
   (e) If the `axes` attribute is not defined, search for the one-dimensional NeXus fields with attribute `primary="1"`.
   (f) These are the dimension scales to label the axes of each dimension of the data.
   (g) Link each dimension scale to the respective data dimension by the `axis` attribute (`axis="1"`, `axis="2"`, ... up to the rank of the data).
   (h) If necessary, close the `NXdata` group, open the next one and repeat steps 3 to 6.
   (i) If necessary, close the `NXentry` group, open the next one and repeat steps 2 to 7.

   Consult the NeXus API section, which describes the routines available to program these operations. In the course of time, generic NeXus browsers will provide this functionality automatically.

5. *Why are the NeXus classes so complicated? I'll never store all that information*

   The NeXus classes are essentially glossaries of terms. If you need to store a piece of information, consult the class definitions to see if it has been defined. If so, use it. However, it is not compulsory to include every item that has been defined if it is not relevant to your experiment. On the other hand, if there is a NeXus definition for your instrument, you are recommended to include all the compulsory items if you want to use standard software to analyze your data.

6. *I want to produce an application definition. How do I go about it?*

   Read the NXDL Tutorial in Section C.4. If you encounter any problems because the classes are not sufficient to describe your configuration, please contact the NIAC Executive Secretary explaining the problem, and post a suggestion at the relevant class wiki page. The NIAC is always willing to consider proposals to amend the base classes. The procedures are defined in the NIAC constitution.[1]

7. *Explain what is the purpose of `NXdata`.*

   The information provided by the `NXdata` group, identification of the default plottable data, is one of the basic motivations (see Section 1.1.1) for the NeXus standard. The choice of the name of the `NXdata` class is historic so renaming it to something more descriptive of its actual function would break legacy data files. `NXdata` contains links to the plottable data stored elsewhere in the `NXentry`.

8. *Can I use a NXDL specification to parse a NeXus data file?*

   This should be possible as there is nothing in the NeXus specifications to prevent this but it is not implemented in NAPI. You would need to implement it for yourself. You would be wise to consult the algorithms in the Java version of `NXvalidate` (see nxvalidate) for more details.

[1] Refer to the most recent version of the NIAC constitution on the NIAC wiki: `http://www.nexusformat.org/NIAC`

Appendix E

Brief history of the NeXus format

June 1994 Mark Koennecke (PSI) made a proposal using netCDF for the European neutron scattering community while working at ISIS

August 1994 Jon Tischler and Mitch Nelson (ORNL) proposed an HDF-based format as a standard for data storage at APS

1995 and 1996 This was the basis for the current designed which was developed at SoftNeSS 1995 (at NIST) and SoftNeSS 1996 (at ANL)

August 1996 NeXus Abstract Programmer Interface (NAPI) released

October 1996 Przemek Klosowski (NCNR) produced a first draft of the NeXus proposal drawing on ideas from both sources

July 1997 SINQ at PSI started writing NeXus files to store raw data

summer 2001 MLNSC at LANL started writing NeXus files to store raw data

September 2002 NeXus API version 2.0.0 is released

October 2003 NeXus International Advisory Committee (NIAC) formed and first meeting held at CalTech

July 2005 NeXus API version 3.0.0 is released

May 2007 NeXus API version 4.0.0 is released

October 2007 NeXus API version 4.1.0 is released

October 2008 The NeXus Definition Language is defined

September 2009 NXDL and draft `NXsas` presented to canSAS at SAS2009 conference

January 2010 NXDL presented to ESRF HDF5 workshop on hyperspectral data

Appendix F

NIAC

The purpose of the NeXus International Advisory Committee (NIAC)[1] is to supervise the development and maintenance of the NeXus common data format for neutron, x-ray, and muon science. This purpose includes, but is not limited to, the following activities.

1. To establish policies concerning the definition, use, and promotion of the NeXus format.
2. To ensure that the specification of the NeXus format is sufficiently complete and clear for its use in the exchange and archival of neutron, x-ray, and muon data.
3. To receive and examine all proposed amendments and extensions to the NeXus format. In particular, to ratify proposed instrument and group class definitions, to ensure that the data structures conform to the basic NeXus specification, and to ensure that the definitions of data items are clear and unambiguous and conform to accepted scientific usage.
4. To ensure that documentation of the NeXus format is sufficient, current, and available to potential users both on the internet and in other forms.
5. To coordinate with the developers of the NeXus Application Programming Interface to ensure that it supports the use of the NeXus format in the neutron, x-ray, and muon communities, and to promote other software development that will benefit users of the NeXus format.
6. To coordinate with other organizations that maintain and develop related data formats to ensure maximum compatibility.

The committee will meet at least once every calendar year according to the following plan:

- In years coinciding with the NOBUGS series of conferences (once every two years), members of the entire NIAC will meet as a satellite meeting to NOBUGS, along with interested members of the community.
- In intervening years, the executive officers of the NIAC will attend, along with interested members of the NIAC. This is intended to be a working meeting with a small group.

[1]For more details about the NIAC constitution, procedures, and meetings, refer to the NIAC wiki page: `http://www.nexusformat.org/NIAC`

Chapter 6

Index