## CONTENTS

1 NeXus: User Manual
   1.1 NeXus Introduction ................................................. 3
   1.2 NeXus Design .................................................... 17
   1.3 Constructing NeXus Files and Application Definitions ................. 60
   1.4 Strategies for storing information in NeXus data files .................. 72
   1.5 Verification and validation of files .................................. 76
   1.6 Frequently Asked Questions ....................................... 76

2 Examples of writing and reading NeXus data files 81
   2.1 Code Examples in Various Languages ................................ 81
   2.2 Visualization tools ............................................... 115
   2.3 Examples for Specific Instruments .................................. 119
   2.4 Other tools to handle NeXus data files ............................... 136

3 NeXus: Reference Documentation 137
   3.1 Introduction to NeXus definitions ................................... 137
   3.2 NXDL: The NeXus Definition Language ................................ 141
   3.3 NeXus Class Definitions ........................................... 170

4 NAPI: NeXus Application Programmer Interface (frozen) 599
   4.1 Status ............................................................. 599
   4.2 Overview .......................................................... 599
   4.3 Core API .......................................................... 600
   4.4 Utility API ......................................................... 607
   4.5 Building Programs ................................................ 608
   4.6 Reporting Bugs in the NeXus API ................................... 609

5 NeXus Community 611
   5.1 NeXus Webpage .................................................... 611
   5.2 Contributed Definitions ............................................ 611
   5.3 Other Ways NeXus Coordinates with the Scientific Community ........ 611

6 Installation 615
   6.1 Precompiled Binary Installation .................................... 615
   6.2 Source Installation ................................................. 616
   6.3 Releases ........................................................... 616

7 NeXus Utilities 619
   7.1 Utilities supplied with NeXus ........................................ 619
   7.2 Validation .......................................................... 620
   7.3 Other Utilities ...................................................... 620
Nexus

https://www.nexusformat.org/


1.1 NeXus Introduction

NeXus\(^1\) is an effort by an international group of scientists *motivated* to define a common data exchange format for neutron, X-ray, and muon experiments. NeXus is built on top of the scientific data format HDF5 and adds domain-specific rules for organizing data within HDF5 files in addition to a dictionary of well-defined domain-specific field names. The NeXus data format has three purposes:

1. *raw data*: NeXus defines a format that can serve as a container for all relevant data associated with a scientific instrument or beamline. This is a very important use case. This includes the case of streaming data acquisition, where time stamped data are logged.

2. *processed data*: NeXus also defines standards for processed data. This is data which has undergone some form of data reduction or data analysis. NeXus allows storing the results of such processing together with documentation about how the processed data was generated.

3. *standards*: NeXus defines standards in the form of *application definitions* for the exchange of data between applications. NeXus provides standards for both raw and processed data.

A community 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 becomes more challenging, individual scientists, or even institutions, find 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 stimulates the design of more sophisticated visualization tools. Additional background information is given in the chapter titled *Brief history of NeXus*.

This section is designed to give a brief introduction to NeXus, the data format and tools that have 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.

---

\(^1\) *J. Appl. Cryst.* (2015), 48, 301-305 (https://doi.org/10.1107/S1600576714027575)
1.1.1 What is NeXus?

The NeXus data format has four components:

**A set of design principles**

to help people understand what is in the data files.

**A set of data storage objects**

(Base Class Definitions and Application Definitions) to allow the development of portable analysis software.

**A set of subroutines**

(Utilities and examples) to make it easy to read and write NeXus data files.

**A Scientific Community**

to provide the scientific data, advice, and continued involvement with the NeXus standard. NeXus provides a forum for the scientific community to exchange ideas in data storage.

In addition, NeXus relies on a set of low-level file formats to actually store NeXus files on physical media. Each of these components are described in more detail in the *Physical File format* section.

The NeXus Application-Programmer Interface (NAPI), which provides the set of subroutines for reading and writing NeXus data files, is described briefly in *NAPI: The NeXus Application Programming Interface*. (Further details are provided in the NAPI chapter.)

The principles guiding the design and implementation of the NeXus standard are described in the NeXus Design chapter.

Base classes, which comprise the data storage objects used in NeXus data files, are detailed in the *Base Class Definitions* chapter.

Additionally, a brief list describing the set of NeXus Utilities available to browse, validate, translate, and visualise NeXus data files is provided in the *NeXus Utilities* chapter.

### A Set of Design Principles

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

**Groups**

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

**Fields**

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). Fields are represented as HDF5 datasets.

**Attributes**

Extra information required to describe a particular group or field, such as the data units, can be stored as a data attribute. Attributes can also be given at the file level of an HDF5 file.

**Links**

Links are used to represent the same information in different places.

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.
Example of a NeXus File

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

Example of a NeXus Data File

![Tree Diagram of NeXus File Structure]

Note that each field is identified by a name, such as `counts`, but each group is identified both by a name and, after a colon as a delimiter, the class type, e.g., `monitor:NXmonitor`. The class types, 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.

Important Classes

Here are some of the important classes found in nearly all NeXus files. A complete list can be found in the NeXus Base Classes chapter. A complete list of all NeXus classes may be found in the NeXus Class Definitions chapter.

**Note:** `NXentry` is the only class required in a valid NeXus data file.

**NXentry**

*Required:* 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 may contain one or more `NXdata` groups. 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 \texttt{NXdata} group. That means it should contain the axis labels and titles as well as the data.

\textbf{NXsample}

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

\textbf{NXinstrument}

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

\begin{figure}[h]
  \centering
  \includegraphics[width=0.5\textwidth]{nexus-excerpt}
  \caption{\texttt{NXinstrument} excerpt}
  \end{figure}

Since an instrument can include several beamline components each defined by several parameters, the components 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.

\textbf{Simple Example}

NeXus data files do not need to be complicated. In fact, the following diagram shows an extremely simple NeXus file (in fact, the simple example shows the minimum information necessary for a NeXus data file) that could be used to transfer data between programs. (Later in this section, we show how to write and read this simple example.)

\begin{figure}[h]
  \centering
  \includegraphics[width=0.5\textwidth]{simple-data-file}
  \caption{Example structure of a simple data file}
  \end{figure}
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. Here is the structure of a 
very simple NeXus data file (examples/verysimple.nx5):

Structure of a very simple NeXus Data file

```python
verysimple.nx5 : NeXus data file
   @default = "entry"
   entry:NXentry
      @NX_class = NXentry
      @default = "data"
      data:NXdata
         @NX_class = NXdata
         @signal = "counts"
         @axes = "two_theta"
         @two_theta_indices = [0]
         counts:int32[15] = [1193, 4474, 53220, '...', 1000]
         @units = "counts"
         @long_name = photodiode counts
         two_theta:float64[15] = [18.9094, 18.9096, '...', 18.9122]
         @units = "degrees"
         @long_name = "two_theta (degrees)"
```

NeXus files are easy to visualize. Here, this data is plotted using NeXPy simply by opening the NeXus data file and 
double-clicking the file name in the list:

Plot of a very simple NeXus HDF5 Data file

NeXus files are easy to create. This example NeXus file was created using a short Python program and the h5py 
package:
Using Python to write a very simple NeXus HDF5 Data file

```python
#!/usr/bin/env python
"uses h5py to build the verysimple.nx5 data file"

import h5py

diode = [1193, 4474, 53220, 274310, 515430, 827880, 1227100, 1434640, 1330280, 1037070, 598720, 316460, 56677, 1000, 1000]

with h5py.File('verysimple.nx5', 'w') as f:
    f.attrs['default'] = 'entry'

    nxentry = f.create_group('entry')
    nxentry.attrs['NX_class'] = 'NXentry'
    nxentry.attrs['default'] = 'data'

    nxdata = nxentry.create_group('data')
    nxdata.attrs['NX_class'] = 'NXdata'
    nxdata.attrs['signal'] = 'counts'
    nxdata.attrs['axes'] = 'two_theta'
    nxdata.attrs['two_theta_indices'] = [0,]

    tth = nxdata.create_dataset('two_theta', data=angle)
    tth.attrs['units'] = 'degrees'
    tth.attrs['long_name'] = 'two_theta (degrees)'

    counts = nxdata.create_dataset('counts', data=diode)
    counts.attrs['units'] = 'counts'
    counts.attrs['long_name'] = 'photodiode counts'
```

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 specialized visualization or analysis software, you will need to know precisely what specific 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.

The set of data storage objects is divided into three parts: base classes, application definitions, and contributed definitions. The base classes represent a set of components that define the dictionary of all possible terms to be used with that component. The application definitions specify the minimum required information to satisfy a particular scientific or data analysis software interest. The contributed definitions have been submitted by the scientific community for incubation before they are adopted by the NIAC or for availability to the community.

These instrument definitions are formalized as XML files, using NXDL, to specify the names of fields, and other NeXus data objects. The following is an example of such a file for the simple NeXus file shown above.
A very simple NeXus Definition Language (NXDL) file

```xml
<?xml version="1.0" ?>
<definition
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"
category="base"
name="NXverysimple"
type="group" extends="NXobject">
  <doc>
    A very simple NeXus NXDL file
  </doc>
  <group type="NXentry">
    <group type="NXdata">
      <field name="counts" type="NX_INT" units="NX_UNITLESS">
        <doc>
          counts recorded by detector
        </doc>
      </field>
      <field name="two_theta" type="NX_FLOAT" units="NX_ANGLE">
        <doc>
          rotation angle of detector arm
        </doc>
      </field>
    </group>
  </group>
</definition>
```

Complete examples of reading and writing NeXus data files are provided later. This chapter has several examples of writing and reading NeXus data files. 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. An example of how to do this is shown in the Creating a NXDL Specification section.

**A Set of Subroutines**

NeXus data files are high-level so the user only needs to know how the data are referenced in the file but does not need to be concerned where the data are stored in the file. Thus, the data are most easily accessed using a subroutine library tuned to the specifics of the data format.

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. 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:
Simple example of reading data using the NeXus API

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

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.

Scientific Community

NeXus began as a group of scientists with the goal of defining a common data storage format to exchange experimental results and to exchange ideas about how to analyze them.

The NeXus Community provides the scientific data, advice, and continued involvement with the NeXus standard. NeXus provides a forum for the scientific community to exchange ideas in data storage.

The NeXus International Advisory Committee (NIAC) supervises the development and maintenance of the NeXus common data format for neutron, X-ray, and muon science through the NeXus class definitions and oversees the maintenance of the NeXus Application Programmer Interface (NAPI) as well as the technical infrastructure.

Representation of data examples

Most of the examples of data files have been written in a format intended to show the structure of the file rather than the data content. In some cases, where it is useful, some of the data is shown. Consider this prototype example:

```plaintext
example of NeXus data file structure
```

```python
entry:NXentry
   instrument:NXinstrument
      detector:NXdetector
         data:[]
            @long_name = "strip detector 1-D array"
            bins:[0, 1, 2, ... 1023]
            @long_name = "bin index numbers"
      sample:NXsample
         name = "zeolite"
      data:NXdata
         @signal = "data"
         @axes = ["bins", "bins"]
         @bins_indices = [0, 1]
         data --> /entry/instrument/detector/data
         bins --> /entry/instrument/detector/bins
```

Some words on the notation:

- Hierarchy is represented by indentation. Objects on the same indentation level are in the same group.
- The combination `name:NXclass` denotes a NeXus group with name `name` and class `NXclass`.
- A simple name (no following class) denotes a field. An equal sign is used to show the value, where this is important to the example.
- Sometimes, a data type is specified and possibly a set of dimensions. For example, `energy:NX_NUMBER[NE]` says `energy` is a 1-D array of numbers (either integer or floating point) of length `NE`.
- Attributes are noted as `@name="value"` pairs. The `@` symbol only indicates this is an attribute and is not part of the attribute name.
- Links are shown with a text arrow `-->` indicating the source of the link (using HDF5 notation listing the sequence of names).

Line 1 shows that there is one group at the root level of the file named `entry`. This group is of type `NXentry` which means it conforms to the specification of the `NXentry` NeXus base class. Using the HDF5 nomenclature, we would refer to this as the `/entry` group.

Lines 2, 8, and 10: The `/entry` group contains three subgroups: `instrument`, `sample`, and `data`. These groups are of type `NXinstrument`, `NXsample`, and `NXdata`, respectively.

Line 4: The data of this example is stored in the `/entry/instrument/detector` group in the dataset called `data` (HDF5 path is `/entry/instrument/detector/data`). The indication of `data:\[]` says that `data` is an array of unspecified dimension(s).

Line 5: There is one attribute of `/entry/instrument/detector/data`: `long_name`. This attribute might be used by a plotting program as the axis title.

Line 6 (reading `bins:\[0, 1, 2, ... 1023\]`) shows that `bins` is a 1-D array of length presumably 1024. A small, representative selection of values are shown.

Line 7: An attribute that shows a descriptive name of `/entry/instrument/detector/bins`. This attribute might be used by a NeXus client while plotting the data.

Line 9 (reading `name = "zeolite"`) shows how a string value is represented.

Line 11 says that the default data to be plotted is called `data`.

Line 12 says that each axis `dimension scale` of `data` is described by the field called `bins`.

Line 13 says that `bins` will be used for axis 0 and axis 1 of `data`.

Lines 14-15: The `/entry/data` group has two datasets that are actually linked as shown to data sets in a different group. (As you will see later, the `NXdata` group enables NeXus clients to easily determine what to offer for display on a default plot.)

### Class path specification

In some places in this documentation, a path may be shown using the class types rather than names. For example:

```
/NXentry/NXinstrument/NXcrystal/wavelength
```

identifies a dataset called `wavelength` that is inside a group of type `NXcrystal`...

As it turns out, this syntax is the syntax used in NXDL `link` specifications. This syntax is also used when the exact name of each group is either unimportant or not specified.

If default names are taken for each class, then the above class path is expressed as this equivalent HDF5 path:

```
/entry/instrument/crystal/wavelength
```

In some places in this documentation, where clarity is needed to specify both the path and class name, you may find this equivalent path:

```
/entry:NXentry/instrument:NXinstrument/crystal:NXcrystal/wavelength
```
Motivations for the NeXus standard in the Scientific Community

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 Könnecke (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 across three workshops organized by Ray Osborn (ANL), SoftNeSS’94 (Argonne Oct. 1994), 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. **Unified format for reduction and analysis**
3. **Defined dictionary of terms**

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` group so that any program that is used to browse NeXus data files can provide a *best representation* without request for user input. A description of how simple plotting is facilitated in NeXus is shown in the section titled *Find the plottable data*.

NeXus is about how to find and annotate the data to be plotted but not to describe how the data is to be plotted. ([https://www.nexusformat.org/NIAC2018Minutes.html#nxdata-plottype-attribute](https://www.nexusformat.org/NIAC2018Minutes.html#nxdata-plottype-attribute))

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 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. This unified format also needs to allow data to be written to storage as efficiently as possible to enable use with high-speed data acquisition.

*Self-description*, combined with a reliance on a *multi-platform* (and thereby *portable*) data storage format, are valued components of a data storage format where the longevity of the data is expected to be longer than the lifetime of the facility at which it is 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 ex-
tendable) 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.

More recently, NeXus has recognized that many facilities face increased performance requirements and support for
writing HDF5 directly in high level languages has become better (for example with h5py for Python). For that reason
HDF5 has become the default recommended storage format for NeXus and the use of the NeXus API for new projects
is no longer encouraged. In NeXus has recently defined encoding of information in ways that are not compatible with
the existing HDF4 and XML container formats (using attribute arrays). The move to HDF5 is strongly advised.

For cases where legacy support of the XML or HDF4 storage backends is required the NeXus API will still be main-
tained though and provide an upgrade path via the utilities to convert between the different backends.

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. NeXus Application Definitions provide the means to document the lexicography for use in data files of
that scientific specialty.

NAPI: The NeXus Application Programming Interface

The NeXus API consists of routines to read and write NeXus data files. It was written to provide a simple to use and
consistent common interface for all supported backends (XML, HDF4 and HDF5) to scientific programmers and other
users of the NeXus Data Standard.

Note: It is not necessary to use the NAPI to write or read NeXus data files. The intent of the NAPI is to simplify the
programming effort to use the HDF programming interface. There are Examples of writing and reading NeXus data
files to help you understand.

This section will provide a brief overview of the available functionality. Further documentation of the NeXus Application
Programming Interface (NAPI) for bindings to specific programming language can be found in the NAPI chapter
and may be downloaded from the NeXus development site.¹

For an even more detailed description of the internal workings of NAPI see the NeXus Internals manual, copied from
the NeXus code repository. That document is written for programmers who want to work on the NAPI itself. If you
are new to NeXus and just want to implement basic file reading or writing you should not start by reading that.

¹ https://github.com/nexusformat/code/releases/
How do I write a NeXus file?

The NeXus Application Program Interface (NAPI) 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, Python, C++, and IDL.

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 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. Following the simple example titled Example structure of a simple data file, we walk through a schematic of NeXus program written in C (without any error checking or real data).

Writing a simple NeXus file using NAPI

Note: We assume the program can define the arrays tth and counts, each length n. This part has been omitted from the example code.

```c
#include "napi.h"

int main()
{
    /* we start with known arrays tth and counts, each length n */
    NXhandle fileID;
    NXopen ("NXfile.nxs", NXACC_CREATE, &fileID);
    NXmakegroup (fileID, "Scan", "NXentry");
    NXopengroup (fileID, "Scan", "NXentry");
    NXmakegroup (fileID, "data", "NXdata");
    NXopengroup (fileID, "data", "NXdata");
    NXmakedata (fileID, "two_theta", NX_FLOAT32, 1, &n);
    NXopendata (fileID, "two_theta");
    NXputdata (fileID, tth);
    NXputattr (fileID, "units", "degrees", 7, NX_CHAR);
    NXclosedata (fileID); /* two_theta */
    NXmakedata (fileID, "counts", NX_FLOAT32, 1, &n);
    NXopendata (fileID, "counts");
    NXputdata (fileID, counts);
    NXclosedata (fileID); /* counts */
    NXclosegroup (fileID); /* data */
    NXclosegroup (fileID); /* Scan */
    NXclose (&fileID);
    return;
}
```
program analysis

1. line 7:
   Open the file NXfile.nxs with create access (implying write access). NAPI\(^2\) returns a file identifier of type NXhandle.

2. line 7:
   Next, we create the NXentry group to contain the scan using NXmakegroup() and then open it for access using NXopengroup().\(^3\)

3. line 10:
   The plottable data is contained within an NXdata group, which must also be created and opened.

4. line 12:
   To create a field, call NXmakedata(), specifying the data name, type (NX_FLOAT32), rank (in this case, 1), and length of the array (n). Then, it can be opened for writing.\(^4\)

5. line 14:
   Write the data using NXputdata().

6. line 15:
   With the field still open, we can also add some field attributes, such as the data units,\(^5\) which are specified as a character string (type="NX_CHAR") that is 7 bytes long.

7. line 16:
   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.

8. line 17:
   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.

9. line 20:
   Finally, close the groups (NXdata and NXentry) before closing the file itself.

How do I read a NeXus file?

Reading a NeXus file works in the same way by traversing the tree with the handle.

This schematic C code will read the two-theta array created in the example above. (Again, compare this example with Reading a simple NeXus file using native HDF5 commands in C.)

---

\(^2\) NAPI: NeXus Application Programmer Interface (frozen)
\(^3\) See the chapter Base Class Definitions for more information.
\(^4\) The NeXus Data Types section describes the available data types, such as NX_FLOAT32 and NX_CHAR.
\(^5\) NeXus Data Units
\(^6\) The NeXus rule about data units is described in the NeXus Data Units section.
\(^7\) see Data Types allowed in NxDL specifications
Reading a simple NeXus file using NAPI

```c
NXopen ('NXfile.nxs', NXACC_READ, &fileID);
NXopengroup (fileID, "Scan", "NXentry");
NXopengroup (fileID, "data", "NXdata");
NXopenedata (fileID, "two_theta");
NXgetinfo (fileID, &rank, dims, &datatype);
NXmalloc ((void **) &tth, rank, dims, datatype);
NXgetdata (fileID, tth);
NXclosedata (fileID);
NXclosegroup (fileID);
NXclosegroup (fileID);
NXclose (fileID);
```

How do I browse a NeXus file?

NeXus files can also be viewed by a command-line browser, `nxbrowse`, which is included as a helper tool in the NeXus API distribution. The following is an example session of nxbrowse to view a data file.

Using nxbrowse

```bash
%> nxbrowse lrcs3701.nxs

NXBrowse 3.0.0. Copyright (C) 2000 R. Osborn, M. Koennecke, P. Klosowski

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)
```

(continues on next page)
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

program analysis

1. line 1:
   Start nxbrowse from the UNIX command line and open file lrcs3701.nxs from IPNS/LRMECS.
2. line 8:
   List the contents of the current group.
3. line 11:
   Open the NeXus group Histogram1.
4. line 23:
   Print the contents of the NeXus data labeled title.
5. line 41:
   Close the current group.
6. line 43:
   Quits nxbrowse.

The source code of nxbrowse\(^8\) provides an example of how to write a NeXus reader. The test programs included in the NeXus API may also be useful to study.

1.2 NeXus Design

This chapter actually defines the rules to use for writing valid NeXus files. An explanation of NeXus objects is followed by the definition of NeXus coordinate systems, the rules for structuring files and the rules for storing single items of data.

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 or an entire multi-component instrument. This flexibility is a necessity as NeXus strives to capture data from a wild variety of applications in X-ray, muSR and neutron scattering. The flexibility is achieved through a hierarchical structure, with related fields 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.

\(^8\) https://github.com/nexusformat/code/blob/master/applications/NXbrowse/NXbrowse.c
1.2.1 NeXus Objects and Terms

Before discussing the design of NeXus in greater detail it is necessary to define the objects and terms used by NeXus. These are:

**Groups**
Levels in the NeXus hierarchy. May contain fields and other groups.

**Fields**
Multidimensional arrays and scalars representing the actual data to be stored

**Attributes**
Attributes containing additional metadata can be assigned to groups, fields, or files.

**Links**
Elements which point to data stored in another place in the file hierarchy

**NeXus Base Classes**
Dictionaries of names possible in the various types of NeXus groups

**NeXus Application Definitions**
Describe the minimum content of a NeXus file for a particular usage case

In the following sections these elements of NeXus files will be defined in more detail.

**Note:** Notation used to describe a NeXus data file

In various places in the NeXus manual, contents of a NeXus data file are described using a tree structure, such as in the *Introduction*.

The tree syntax is a very condensed version (with high information density) meant to convey the structure of the HDF file.

- Groups have a `/` appended to their name (with NeXus class name shown)
- Indentation shows membership in the lesser indented parent above.
- Fields have a data type and value appended (for arrays, this may be an abbreviated view)
- Attributes (of groups or fields) are prefixed with `@`.
- NeXus-style links are described with some sort of arrow notation such as `-->`.

**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 but they must have different names (based on the HDF rules).

For the class names used with NeXus data groups the prefix `NX` is reserved. Thus all NeXus class names start with `NX`. 
Fields

Fields (also called data fields, data items or data sets) 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). Fields are identified by their names, which must be unique within the group in which they are stored. Some fields have engineering units to be specified. In some cases, such as `/NXdata/DATA`, a field is expected to have be an array of several dimensions.

Examples of fields

- **variable** (*NX_NUMBER*)
  Dimension scale defining an axis of the data.

- **variable_errors** (*NX_NUMBER*)
  Errors (uncertainties) associated with axis variable.

- **wavelength** (*NX_FLOAT*)
  Wavelength of radiation, units="NX_FLOAT"

- **chemical_formula** (*NX_CHAR*)
  The chemical formula specified using CIF conventions.

- **name** (*NX_CHAR*)
  Name of user responsible for this entry.

- **data** (*NX_NUMBER*)
  Data values from the detector, units="NX_ANY"

See the sections [Data Types allowed in NXDL specifications](#) and [Unit Categories allowed in NXDL specifications](#) for complete lists of the data types and engineering units types, respectively.

In the case of streaming data acquisition, when time-stamped values of data are collected, fields can be replaced with `NXlog` structures of the same name. For example, if time stamped data for wavelength is being streamed, wavelength would not be an array but a `NXlog` structure.

Attributes

Attributes are extra (meta-)information that are associated with particular groups or fields. They are used to annotate 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. In a tree structure, an attribute is usually shown with a `@` prefix, such as `@units`. A description of some of the many possible attributes can be found in the next table:

Examples of attributes

- **units** (*NX_CHAR*)
  Data units given as character strings, must conform to the NeXus units standard. See the [NeXus Data Units](#) section for details.

- **signal** (*NX_CHAR*)
  Defines which data set contains the signal to be plotted. Use `signal="{dataset_name}"` where `{dataset_name}` is the name of a field (or link to a field) in the `NXdata` group. The field referred to by the `signal` attribute might be referred to as the “signal data”.

- **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

data_offset (NX_INT)
  Rank values of offsets to use for each dimension if the data is not in C storage order

interpretation (NX_CHAR)
  Describes how to display the data. rgba, hsla and cmyk are \((n \times m \times 4)\) arrays, where the 4 channels are the colour channels appropriately. If the image data does not contain an alpha channel, then the array should simply be \((n \times m \times 3)\). Allowed values include:
  - scalar (0-D data)
  - scaler DEPRECATED, use scalar
  - spectrum (1-D data)
  - image (2-D data)
  - rgb-image (3-D data)
  - rgba-image (3-D data)
  - hsl-image (3-D data)
  - hsla-image (3-D data)
  - cmyk-image (3-D data)
  - vertex (3-D data)

File attributes

Finally, some attributes are defined at file level. They are specified in the base class \(NXroot\).

Links

Python h5py code to make NeXus links

The section titled HDF5 in Python provides example python code to create links (both internal and external) in NeXus data files. See the routines:

- \{(hdf5_object).\_id.link()\
- h5py.ExternalLink()

Links are pointers to existing data somewhere else. The concept is very much like symbolic links in a unix filesystem. The NeXus definition sometimes requires to have access to the same data in different groups in the same file. For example: detector data is stored in the NXinstrument/NXdetector group but may be needed in NXdata for automatic plotting. Rather then replicating the data, NeXus uses links in such situations. See the figure for a more descriptive representation of the concept of linking.
NeXus links are HDF5 hard links with an additional target attribute. The target attribute is added\(^1\) for NeXus to distinguish the HDF5 path to the original\(^2\) dataset. The value of the target attribute is the HDF5 path\(^3\) to the original dataset.

NeXus links are best understood with an example. The canonical location (expressed as a NeXus class path) to store wavelength (see Strategies: The wavelength) has been:

\[
/NXentry/NXinstrument/NXcrystal/wavelength
\]

An alternative location for this field makes sense to many, especially those not using a crystal to create monochromatic radiation:

\[
/NXentry/NXinstrument/NXmonochromator/wavelength
\]

These two fields might be hard linked together in a NeXus data file (using HDF5 paths such /entry/instrument):

---

1. When using the NAPI, the target attribute is added automatically. When the NAPI is not used to write NeXus/HDF5 files, this attribute must be added. Here are the steps to follow:
   1. Get the HDF5 reference ID of the source item (field, group, or link) to be linked.
   2. If the ID does not have a target attribute defined: #. Get the absolute HDF5 address\(^1\) of the ID. #. Create a target attribute for the ID. #. Set the target attribute’s value to the absolute HDF5 address of the ID.
   3. Create an HDF5 hard link\(^4\) to the ID at the desired (new) HDF5 address.

3. When using the target attribute, always specify the HDF5 address as an absolute* address (starts from the HDF5 root, such as: /entry/instrument/detector/polar_angle) rather than a relative address (starting from the current group, such as: detector/polar_angle).

---

Note: The target attribute does not work for external file links. The NIAC is working at resolving the technical limitations

---

4. HDF5 hard link: [https://portal.hdfgroup.org/display/HDF5/H5L_CREATE_HARD](https://portal.hdfgroup.org/display/HDF5/H5L_CREATE_HARD)

2. The notion of an original dataset with regard to links is a NeXus abstraction. In truth, HDF5 makes no distinction which is the original dataset. But, when the file is viewed with a tool such as h5dump, confusion often occurs over which dataset is original and which is a link to the original. Actually, both HDF5 paths point to the exact, same dataset which exists at a specific offset in the HDF5 file.

   See the [Frequently Asked Questions](https://portal.hdfgroup.org/display/HDF5/H5L_CREATE_HARD) question: I’m using links to place data in two places. Which one should be the data and which one is the link?
It is possible that the linked field or group has a different name than the original. One obvious use of this capability is to adapt to a specific requirement of an application definition. For example, suppose some application definition required the specification of wavelength as a field named `lambda` in the entry group. This requirement can be satisfied easily:

```
entry:NXentry
    ...
    instrument:NXinstrument
        ...
        crystal:NXcrystal
        ...
        wavelength:NX_FLOAT = 154.
            @target="/entry/instrument/crystal/wavelength"
            @units="pm"
        ...
        monochromator:NXmonochromator
            ...
        wavelength --> "/entry/instrument/crystal/wavelength"
        ...
        lambda --> "/entry/instrument/crystal/wavelength"
```

**External File Links**

NeXus also allows for links to external files. Consider the case where an instrument uses a detector with a closed-system software support provided by a commercial vendor. This system writes its images into a NeXus HDF5 file. The instrument’s data acquisition system writes instrument metadata into another NeXus HDF5 file. In this case, the instrument metadata file might link to the data in the detector image file. Here is an example (from Diamond Light Source) showing an external file link in HDF5:

```
entry:NXentry
    ...
    instrument:NXinstrument
        ...
        crystal:NXcrystal
        ...
        wavelength:NX_FLOAT = 154.
            @target="/entry/instrument/crystal/wavelength"
            @units="pm"
        ...
        monochromator:NXmonochromator
            ...
        wavelength --> "/entry/instrument/crystal/wavelength"
```

It is possible that the linked field or group has a different name than the original. One obvious use of this capability is to adapt to a specific requirement of an application definition. For example, suppose some application definition required the specification of wavelength as a field named `lambda` in the entry group. This requirement can be satisfied easily:

```
entry:NXentry
    ...
    instrument:NXinstrument
        ...
        crystal:NXcrystal
        ...
        wavelength:NX_FLOAT = 154.
            @target="/entry/instrument/crystal/wavelength"
            @units="pm"
        ...
        monochromator:NXmonochromator
            ...
        wavelength --> "/entry/instrument/crystal/wavelength"
        ...
        lambda --> "/entry/instrument/crystal/wavelength"
```
Example of linking to data in an external HDF5 file

```
EXTERNAL_LINK "data" {
    TARGETFILE "/dls/i22/data/2012/sm7594-1/i22-69201-Pilatus2M.h5"
    TARGETPATH "entry/instrument/detector/data"
}
```

**Note:** The NAPI code\(^5\) makes no target attribute assignment for links to external files. It is best to avoid using the target attribute with external file links. The NIAC is working at resolving the technical limitations.

The NAPI maintains a group attribute `@napimount` that provides a URL to a group in another file. More information about the `@napimount` attribute is described in the *NeXus Programmers Reference*.\(^6\)

**Combining NeXus links and External File Links**

Consider the case described in *Links to Data in External HDF5 Files*, where numerical data are provided in two different HDF5 files and a master NeXus HDF5 file links to the data through external file links. HDF5 will not allow hard links to be constructed with these data objects in the master file. An error such as *Interfile hard links are not allowed* (as generated from h5py) will arise. This makes sense since there is no such data object in the file.

Instead, it is necessary to make an external file link at each place in the master where external data is to be represented.

**NeXus Base Classes**

Data groups often describe objects in the experiment (monitors, detectors, monochromators, etc.), so that the contents (both fields and/or other 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. This is each data group is identified by a name and a class. The group class, 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. All NeXus class types adopted by the NIAC must begin with `NX`. Classes not adopted by the NIAC must not start with `NX`.

**Note:** NeXus base classes are the components used to build the NeXus data structure.

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.

**Note:** The groups, fields, links, and attributes of a base class definition are all optional, with a few particular exceptions in `NXentry` and `NXdata`. They are named in the specification to describe the exact spelling and usage of the term when it appears.

NeXus base classes are not proper classes in the same sense as used in object oriented programming languages. In fact the use of the term classes is actually misleading but has established itself during the development of NeXus. NeXus base classes are rather dictionaries of field names and their meanings which are permitted in a particular NeXus group implementing the NeXus class. This sounds complicated but becomes easy if you consider that most NeXus groups

\(^5\) `NX5nativeexternallink()`  [https://github.com/nexusformat/code/blob/fe8ddd287ee33961982931e2016cc25f76f95edd/src/napi5.c#L2248](https://github.com/nexusformat/code/blob/fe8ddd287ee33961982931e2016cc25f76f95edd/src/napi5.c#L2248)
\(^6\) [https://manual.nexusformat.org/_static/NeXusIntern.pdf](https://manual.nexusformat.org/_static/NeXusIntern.pdf)
describe instrument components. Then for example, a NXmonochromator base class describes all the possible field
names which NeXus allows to be used to describe a monochromator.

Most NeXus base classes represent instrument components. Some are used as containers to structure information in a
file (NXentry, NXcollection, NXinstrument, NXprocess, NXparameters). But there are some base classes which
have special uses which need to be mentioned here:

**NXdata**

NXdata is used to identify the default plottable data. The notion of a default plot of data is a basic motivation of
NeXus. (see Simple plotting)

**NXlog**

NXlog is used to store time stamped data like the log of a temperature controller. Basically you give a start time,
and arrays with a difference in seconds to the start time and the values read.

**NXcollection**

NXcollection is used to gather together any set of terms. Anything (groups, fields, or attributes) placed in an
NXcollection group will not be validated. One use is to use this as a container class for the various control
system variables from a beamline or instrument.

**NXnote**

This group provides a place to store general notes, images, video or whatever. A mime type is stored together
with a binary blob of data. Please use this only for auxiliary information, for example an image of your sample,
or a photo of your boss.

**NXtransformations**

NXtransformations is used to gather together any set of movable or fixed
elements positioning the device described by the class that contains this. Supercedes NXgeometry.

**NXgeometry** *(superseded by NXtransformations, Page 24, 7)*

NXgeometry and its subgroups NXtranslation, NXorientation, NXshape are used to store abso-
lute positions in the laboratory coordinate system or to define shapes.

These groups can appear anywhere in the NeXus hierarchy, where needed. Preferably close to the component they
annotate or in a NXcollection. All of the base classes are documented in the reference manual.

**NXdata Facilitates Automatic Plotting**

The most notable special base class (or group in NeXus) is NXdata. NXdata is the answer to a basic motivation of
NeXus to facilitate automatic plotting of data. NXdata is designed to contain the main dataset and its associated
dimension scales (axes) of a NeXus data file. The usage scenario is that an automatic data plotting program just opens
a NXentry and then continues to search for any NXdata groups. These NXdata groups represent the plottable data.
An algorithm for identifying the default plottable data is presented in the chapter titled Rules for Storing Data Items in
NeXus Files.

---

7 see: https://github.com/nexusformat/definitions/issues/397
Where to Store Metadata

There are many ways to store metadata about your experiments. Already there are many fields in the various base
classes to store the more common or general metadata, such as wavelength. (For wavelength, see the Strategies: The
wavelength section.)

One common scheme is to store the metadata all in one group. If the group is to be validated for content, then there are
several possibilities, as shown in the next table:

<table>
<thead>
<tr>
<th>base class</th>
<th>intent</th>
</tr>
</thead>
<tbody>
<tr>
<td>NXnote</td>
<td>to store additional information</td>
</tr>
<tr>
<td>NXlog</td>
<td>information that is time-stamped</td>
</tr>
<tr>
<td>NXparameters</td>
<td>parameters for processing or analysis</td>
</tr>
<tr>
<td>NXcollection</td>
<td>to store any unvalidated content</td>
</tr>
</tbody>
</table>

If the content of the metadata group is to be excluded from validation, then store it in a NXcollection group.

NeXus Application Definitions

The objects described so far provide us with the means to store data from a wide variety of instruments, simulations, or
processed data as resulting from data analysis. But NeXus strives to express strict standards for certain applications of
NeXus, too. The tool which NeXus uses for the expression of such strict standards is the NeXus Application Definition.
A NeXus Application Definition describes which groups and data items have to be present in a file in order to prop-
erly describe an application of NeXus. For example for describing a powder diffraction experiment. An application
definition may also declare terms which are optional in the data file. Typically an application definition will contain
only a small subset of the many groups and fields defined in NeXus. NeXus application definitions are also expressed
in the NeXus Definition Language (NXDL). A tool exists which allows one to validate a NeXus file against a given
application definition.

Note: NeXus application definitions define the minimum required information necessary to satisfy data analysis or
other data processing.

Another way to look at a NeXus application definition is as a contract between a file producer (writer) and a file
consumer (reader).

The contract reads: If you write your files following a particular NeXus application definition, I can process
these files with my software.

Yet another way to look at a NeXus application definition is to understand it as an interface definition between data files
and the software which uses this file. Much like an interface in the Java or other modern object oriented programming
languages.

In contrast to NeXus base classes, NeXus supports inheritance in application definitions.

Please note that a NeXus Application Definition will only define the bare minimum of data necessary to perform
common analysis with data. Practical files will nearly always contain more data. One of the beauties of NeXus is that
it is always possible to add more data to a file without breaking its compliance with its application definition.
1.2.2 NeXus Geometry

NeXus supports description of the shape, position and orientation of objects in *The NeXus Coordinate System*. Position and orientation can be defined as *Coordinate Transformations* using the *NXtransformations* class. *Shape Descriptions* use the *NXoff_geometry* or *NXcylindrical_geometry* class.

You may come across old files which use *Legacy Geometry Descriptions*.

**The NeXus Coordinate System**

The NeXus coordinate system is shown *below*. Note that it is the same as that used by *McStas* ([http://mcstas.org](http://mcstas.org)). This choice is arbitrary and any other choice should be possible as long as it is used consistently and application code that reads NeXus files does not assume any prior knowledge of the chosen coordinate system.

![NeXus coordinate system](image)

**Fig. 2: NeXus coordinate system, as viewed from detector**

*Note:* The NeXus definition of +z is opposite to that in the IUCr International Tables for Crystallography, volume G.

**Coordinate Transformations**

In the recommended way of dealing with geometry NeXus uses a series of transformations to place objects in space. In this world view, the absolute position of a component or a detector pixel with respect to the laboratory coordinate system is calculated by applying a series of translations and rotations. Thus a rotation or translation operation transforms the whole coordinate system and gives rise to a new local coordinate system. These transformations between coordinate systems are mathematical operations and can be expressed as matrices and their combination as matrix multiplication. A very important aspect is that the order of application of the individual operations *does* matter. The mathematics behind this is well known and used in such applications such as industrial robot control, flight dynamics and computer games. The beauty in this comes from the fact that the operations to apply map easily to instrument settings and constants. It is also easy to analyze the contribution of each individual operation: this can be studied under the condition that all other operations are at a zero setting.

In order to use coordinate transformations, several pieces of information need to be known:

**Type**

The type of operation: rotation or translation
Direction
The direction of the translation or the direction of the rotation axis

Value
The angle of rotation or the length of the translation

Order
The order of operations to apply to move a component into its place.

Coordinate Transformation Field And Attributes

NeXus chooses to encode information about each transformation as a field in an NXtransformations group in the following way:

value
This is represented in the actual data of the field or the value of the transformation. Its actual name should relate to the physical device used to effect the transformation.

The coordinate transformation attributes are:

transformation_type
This specifies the type of transformation and is either rotation or translation and describes the kind of operation performed

vector (NX_NUMBER)
This is a set of 3 values forming a unit vector for direction that describes the components of either the direction of the rotation axis or the direction along which the translation happens.

offset (NX_NUMBER)
This is a set of 3 values forming the offset vector for a translation to apply before applying the operation of the actual transformation. Without this offset attribute, additional virtual translations would need to be introduced in order to encode mechanical offsets in the axis.

depends_on
The order is encoded through this attribute. The value is the name of the transformation upon which the current transformation depends on.

As each transformation represents possible motion by a physical device, this dependency expresses the attachment order; thus, the current device is attached to (or mounted on) the next device referred to by the attribute.

Allowed values for depends_on are:

- A dot ends the depends_on chain

name
The name of a field within the enclosing group

dir/name
The name of a field further along the path

/dir/dir/name
An absolute path to a field in another group

In addition, for each beamline component, there is a depends_on attribute that points to the field at the head of the axis dependency chain. For example, consider an eulerian cradle as used on a four-circle diffractometer. Such a cradle has a dependency chain
of phi:chi:rotation_angle. Then the depends_on field in $NXsample$ would have 
the value phi.

**NeXus Transformation encoding**

Transformation encoding for an eulerian cradle on a four-circle diffractometer

```
sample: NXsample
   transforms: NXtransformations
      rotation_angle
         @transformation_type=rotation
         @vector=0,1,0
         @offset=0,0,0
         @depends_on=.
      chi
         @transformation_type=rotation
         @vector=0,0,1
         @offset=0,0,0
         @depends_on=rotation_angle
      phi
         @transformation_type=rotation
         @vector=0,1,0
         @offset=0,0,0
         @depends_on=chi
      depends_on
      transforms/phi
```

The type and direction of the NeXus standard operations is documented below in the table: **Actions of standard NeXus fields**. The rule is to always give the attributes to make perfectly clear how the axes work. The CIF scheme also allows 
to store and use arbitrarily named axes in a NeXus file.

The CIF scheme (see $NXtransformations$) is the preferred method for expressing geometry in NeXus.

**Actions of standard NeXus fields**

**Transformation Actions**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>transformation_type</th>
<th>vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>polar_angle</td>
<td>rotation</td>
<td>0 1 0</td>
</tr>
<tr>
<td>azimuthal_angle</td>
<td>rotation</td>
<td>0 0 1</td>
</tr>
<tr>
<td>meridional_angle</td>
<td>rotation</td>
<td>1 0 0</td>
</tr>
<tr>
<td>distance</td>
<td>translation</td>
<td>0 0 1</td>
</tr>
<tr>
<td>height</td>
<td>translation</td>
<td>0 1 0</td>
</tr>
<tr>
<td>x_translation</td>
<td>translation</td>
<td>1 0 0</td>
</tr>
<tr>
<td>chi</td>
<td>rotation</td>
<td>0 0 1</td>
</tr>
<tr>
<td>phi</td>
<td>rotation</td>
<td>0 1 0</td>
</tr>
</tbody>
</table>

For the NeXus spherical coordinate system (described in the legacy section below), the order is implicit and is given 
in the next example.
**implicit order of NeXus spherical coordinate system**

azimuthal_angle:polar_angle:distance

This is also a nice example of the application of transformation matrices:

1. You first apply azimuthal_angle as a rotation around z. This rotates the whole coordinate out of the plane.
2. Then you apply polar_angle as a rotation around y in the tilted coordinate system.
3. This also moves the direction of the z vector. Along which you translate the component to place by distance.

**Shape Descriptions**

**NXoff_geometry**

The shape of instrument components can be described using the **NXoff_geometry** class. **NXoff_geometry** is a polygon-based description, based on the open OFF format. Conversion between OFF files and the NeXus description is straightforward. This is beneficial as existing tools can use, view or manipulate the geometry in OFF files. CAD software, for example FreeCAD, can be used to define the geometry. 3D rendering tools such as Geomview can be used to view the geometry. McStas can use OFF files to define the shape of components for scattering simulations.

The example OFF file shown below defines a cube. The first line containing numbers defines: the number of vertices, the number of faces (polygons) making up the model’s surface, and the number of edges in the mesh. Note, the number of edges must be present but does not need to be correct (http://www.geomview.org/docs/html/OFF.html).

```
1 OFF
2 # cube.off
3 # A cube
4
5 8 6 12
6 1.0 0.0 1.0
7 0.0 1.0 1.0
8 -1.0 0.0 1.0
9 0.0 -1.0 1.0
10 1.0 0.0 0.0
11 0.0 1.0 0.0
12 -1.0 0.0 0.0
13 0.0 -1.0 0.0
14 4 0 1 2 3
15 4 7 4 0 3
16 4 4 5 1 0
17 4 5 6 2 1
18 4 3 2 6 7
19 4 6 5 4 7
```

Following the initial line are the xyz coordinates of each vertex. Proceeding which is the list of faces. Each line defining a face starts with the number of vertices in that face followed by the sequence number of the composing vertices, indexed from zero. The vertex indices form a winding order by defining the face normal by the right-hand rule. The number of vertices in each face need not be constant; a mesh can comprise of polygons of many different orders.

The list of vertices in an OFF file maps directly to the vertices dataset in the **NXoff_geometry** class. The vertex indices of the face list in the OFF file occupy the winding_order dataset of the NeXus class, however the list is flattened to 1D in order to avoid a ragged-edged dataset, which are not easy to work with using HDF libraries. A faces dataset
contains the position of the first entry in winding_order for each face. The NXoff_geometry equivalent of the OFF cube example is shown below.

```plaintext
shape : NXoff_geometry
@NX_class = "NXoff_geometry"
vertices =
  1.0 , 0.0 , 1.0
  0.0 , 1.0 , 1.0
  1.0 , 0.0 , 1.0
  0.0 , 0.0 , 1.0
  0.0 , 0.0 , 0.0
  0.0 , 1.0 , 0.0
  0.0 , 0.0 , 0.0
  0.0 , 0.0 , 0.0
  1.0 , 0.0 , 0.0
  0.0 , -1.0, 0.0
  0.0 , 1.0, -1.0
  1.0 , 0.0, 1.0
  0.0 , 1.0, 0.0
  0.0 , 0.0, 1.0
  0.0 , -1.0, 0.0
  1.0 , 0.0, 0.0
  0.0 , 0.0, 0.0
faces =
  0, 4, 8, 12, 16, 20
winding_order =
  0, 1, 2, 3, 7, 4, 0, 3, 4, 5, 1, 0, 5, 6, 2, 1, 3, 2, 6, 7, 6, 5, 4, 7
```

**NXcylindrical_geometry**

Although the polygon-based description of NXoff_geometry is very flexible, it is not ideal for curved shapes when high precision is required since a very large number of vertices may be necessary. A common example of this is when describing helium tube, neutron detectors. NXcylindrical_geometry provides a more concise method of defining shape for such cases.

Like NXoff_geometry, NXcylindrical_geometry contains a vertices dataset. The indices of three vertices (A, B, C in Cylinder definition with three vertices) in the vertices dataset are used to define each cylinder in the cylinders dataset.

![Cylinder definition with three vertices](image)

**Detector Shape Descriptions**

An NXoff_geometry or NXcylindrical_geometry group named detector_shape can be placed in an NXdetector or NXdetector_module to define the complete shape of the detector. Alternatively, the group can be named pixel_shape and define the shape of a single pixel. In this case, x_pixel_offset, y_pixel_offset and z_pixel_offset datasets of the NXdetector define how the pixel shape is tiled to form the geometry of the complete detector.
Legacy Geometry Descriptions

The above system of chained transformations is the recommended way of encoding geometry going forward. This section describes the traditional way this was handled in NeXus, which you may find occasionally in old files.

Coordinate systems in NeXus have undergone significant development. Initially, only motor positions of the relevant motors were stored without further standardization. This soon proved to be too little and the NeXus polar coordinate system was developed. This system still is very close to angles that are meaningful to an instrument scientist but allows to define general positions of components easily. Then users from the simulation community approached the NeXus team and asked for a means to store absolute coordinates. This was implemented through the use of the NXgeometry class on top of the McStas system. We soon learned that all the things we do can be expressed through the McStas coordinate system. So it became the reference coordinate system for NeXus. NXgeometry was expanded to allow the description of shapes when the demand came up. Later, members of the CIF team convinced the NeXus team of the beauty of transformation matrices and NeXus was enhanced to store the necessary information to fully map CIF concepts. Not much had to be changed though as we choose to document the existing angles in CIF terms. The CIF system allows to store arbitrary operations and nevertheless calculate absolute coordinates in the laboratory coordinate system. It also allows to convert from local, for example detector coordinate systems, to absolute coordinates in the laboratory system.

McStas and NXgeometry System

As stated above, NeXus uses the McStas coordinate system (http://mcstas.org) as its laboratory 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. See below for a drawing of the McStas coordinate system. The origin of this coordinate system is the sample position or, if this is ambiguous, the center of the sample holder with all angles and translations set to zero. The McStas coordinate system is illustrated in the next figure:

![Fig. 4: The McStas Coordinate System](image)

The NeXus NXgeometry class directly uses the McStas coordinate system. NXgeometry classes can appear in any component in order to specify its position. The suggested name to use is geometry. In NXgeometry the NXtranslation/values field defines the absolute position of the component in the McStas coordinate system. The NXorientation/value field describes the orientation of the component as a vector of in the McStas coordinate system.
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. At each component there is local spherical coordinate system with the angles *polar_angle* and *azimuthal_angle*. The size of the sphere is the distance to the previous component.

In order to understand this spherical polar coordinate system it is helpful to look initially at the common condition that *azimuthal_angle* is zero. This corresponds to working directly in the horizontal scattering plane of the instrument. In this case *polar_angle* maps directly to the setting commonly known as *two theta*. Now, there are instruments where components live outside of the scattering plane. Most notably detectors. In order to describe such components we first apply the tilt out of the horizontal scattering plane as the *azimuthal_angle*. Then, in this tilted plane, we rotate to the component. The beauty of this is that *polar_angle* is always *two theta*. Which, in the case of a component out of the horizontal scattering plane, is not identical to the value read from the motor responsible for rotating the component. This situation is shown in *Polar Coordinate System*.

![Polar Coordinate System](image)

**Fig. 5: NeXus Simple (Spherical Polar) Coordinate System**

1.2.3 Rules and Underlying File Formats

Rules for Structuring Information in NeXus Files

All NeXus files contain one or many groups of type *NXentry* at root level. Many files contain only one *NXentry* group, then the name is *entry*. The NXentry level of hierarchy is there to support the storage of multiple related experiments in one file. Or to allow the NeXus file to serve as a container for storing a whole scientific workflow from data acquisition to publication ready data. Also, *NXentry* class groups can contain raw data or processed data. For files with more than one *NXentry* group, since HDF requires that no two items at the same level in an HDF file may have the same name, the NeXus fashion is to assign names with an incrementing index appended, such as *entry1*, *entry2*, *entry3*, etc.

In order to illustrate what is written in the text, example hierarchies like the one in figure *Raw Data* are provided.
Content of a Raw Data NXentry Group

An example raw data hierarchy is shown in figure Raw Data (only showing the relevant parts of the data hierarchy). In the example shown, the data field in the NXdata group is linked to the 2-D detector data (a 512x512 array of 32-bit integers). The attribute signal = data on the NXdata group marks this field as the default plottable data of the data:NXdata group. The NXdata group attribute axes = . . declares that both dimensions of the data field do not have associated dimension scales (plotting routines should use integer scaling for each axis). Note that [,] represents a 2D array.

NeXus Raw Data Hierarchy

An NXentry describing raw data contains at least a NXsample, one NXmonitor, one NXdata and a NXinstrument group. It is good practice to use the names sample for the NXsample group, control for the NXmonitor group holding the experiment controlling monitor and instrument for the NXinstrument group. The NXinstrument group contains further groups describing the individual components of the instrument as appropriate.

The NXdata group contains links to all those data items in the NXentry hierarchy which are required to put up a default plot of the data. As an example consider a SAXS instrument with a 2D detector. The NXdata will then hold a link to the detector image. If there is only one NXdata group, it is good practice to name it data. Otherwise, the name of the detector bank represented is a good selection.

Content of a processed data NXentry group

Processed data, see figure Processed Data, in this context means the results of a data reduction or data analysis program. Note that [] represents a 1D array.

NeXus Processed Data Hierarchy

An entry:NXentry...
NeXus stores such data in a simplified \texttt{NXentry} structure. A processed data \texttt{NXentry} has at minimum a \texttt{NXsample}, a \texttt{NXdata} and a \texttt{NXprocess} group. Again the preferred name for the \texttt{NXsample} group is \texttt{sample}. In the case of processed data, the \texttt{NXdata} group holds the result of the processing together with the associated axis data. The \texttt{NXprocess} group holds the name and version of the program used for this processing step and further \texttt{NXparameters} groups. These groups ought to contain the parameters used for this data processing step in suitable detail so that the processing step can be reproduced.

Optionally a processed data \texttt{NXentry} can hold a \texttt{NXinstrument} group with further groups holding relevant information about the instrument. The preferred name is again \texttt{instrument}. Whereas for a raw data file, NeXus strives to capture as much data as possible, a \texttt{NXinstrument} group for processed data may contain a much-reduced subset.

\textbf{\texttt{NXsubentry} or Multi-Method Data}

Especially at synchrotron facilities, there are experiments which perform several different methods on the sample at the same time. For example, combine a powder diffraction experiment with XAS. This may happen in the same scan, so the data needs to be grouped together. A suitable \texttt{NXentry} would need to adhere to two different application definitions. This leads to name clashes which cannot be resolved easily. In order to solve this issue, the following scheme was implemented in NeXus:

- The complete beamline (all data) is stored in an appropriate hierarchy in an \texttt{NXentry}.
- The \texttt{NXentry} group contains further \texttt{NXsubentry} groups, one for each method.
- Each \texttt{NXsubentry} group is constructed like a \texttt{NXentry} group. It contains links to all those data items required to fulfill the application definition for the particular method it represents.
- The name of the application definition is stored in the \texttt{definition} field of the \texttt{NXsubentry} group.
- Each \texttt{NXsubentry} group contains a \texttt{NXdata} group describing the default plottable data for that experimental method. To satisfy the NeXus requirement of finding the default plottable data from a \texttt{NXentry} group, the \texttt{NXdata} group from one of these \texttt{NXsubentry} groups (the fluorescence data) was linked.

See figure \textit{NeXus Multi Method Hierarchy} for an example hierarchy. Note that \texttt{[,]} represents a 2D array.

\textbf{NeXus Multi Method Hierarchy}

```
entry: NXentry
  @default = data
  user: NXuser
  sample: NXsample
  instrument: NXinstrument
    SASdet: NXdetector
      data: [,]
    fluordet: NXdetector
      data: [,]
    large_area: NXdetector
      data: [,]
  SAS: NXsubentry
```


Rules for Special Cases

Scans

Scans are difficult to capture because they have great variety. Basically, any variable can be scanned. Such behaviour cannot be captured in application definitions. Therefore NeXus solves this difficulty with a set of rules. In this section, \( NP \) is used as a symbol for the number of scan points.

- The scan dimension \( NP \) is always the first dimension of any multi-dimensional dataset. The reason for this is that HDF allows the first dimension of a dataset to be unlimited. Which means, that data can be appended to the dataset during the scan.
- All data is stored as arrays of dimensions \( NP \), original dimensions of the data at the appropriate position in the NXentry hierarchy.
- The NXdata group has to contain links to all variables varied during the scan and the detector data. Thus the NXdata group mimics the usual tabular representation of a scan.
- The NXdata group has attributes to enable the default plotting, as described in the section titled **NXdata Facilitates Automatic Plotting**.

Simple scan

Examples may be in order here. Let us start with a simple case, the sample is rotated around its rotation axis and data is collected in a single point detector. See figure *Simple Scan* for an overview. Then we have:

- A dataset at `NXentry/NXinstrument/NXdetector/data` of length \( NP \) containing the count data.
- A dataset at `NXentry/NXsample/rotation_angle` of length \( NP \) containing the positions of `rotation_angle` at the various steps of the scan.
- `NXdata` contains links to:
  - `NXentry/NXinstrument/NXdetector/data`
  - `NXentry/NXsample/rotation_angle`
- All other fields have their normal dimensions.
NeXus Simple Scan Example

```plaintext
entry:NXentry
    @default = data
    instrument:NXinstrument
        detector:NXdetector
            data[NP]
        sample:NXsample
            rotation_angle[NP]
    control:NXmonitor
        data[NP]
    data:NXdata
        @signal = "data"
        @axes = "rotation_angle"
        @rotation_angle_indices = 0
        data --> /entry/instrument/detector/data
        rotation_angle --> /entry/sample/rotation_angle
```

Simple scan with area detector

The next example is the same scan but with an area detector with `xsize` times `ysize` pixels. The only thing which changes is that `/NXentry/NXinstrument/NXdetector/data` will have the dimensions `NP`, `xsize`, `ysize`. See figure `Simple Scan with Area Detector` for an overview.

NeXus Simple Scan Example with Area Detector

```plaintext
entry:NXentry
    instrument:NXinstrument
        detector:NXdetector
            data:[NP,xsize,ysize]
        sample:NXsample
            rotation_angle[NP]
    control:NXmonitor
        data[NP]
    data:NXdata
        @signal = "data"
        @axes = ["rotation_angle", ".", "."]
        @rotation_angle_indices = 0
        data --> /entry/instrument/detector/data
        rotation_angle --> /entry/sample/rotation_angle
```

The `NXdata` group attribute `axes = rotation_angle . .` declares that only the first dimension of the plottable data has a dimension scale (by name, `rotation_angle`). The other two dimensions have no associated dimension scales and should be plotted against integer bin numbers.
Complex \textit{hkl} scan

The next example involves a complex movement along the \textit{h} axis in reciprocal space which requires multiple motors of a four-circle diffractometer to be varied during the scan. We then have:

- A dataset at \texttt{NXentry/NXinstrument/NXdetector/data} of length \texttt{NP} containing the count data.
- A dataset at \texttt{NXentry/NXinstrument/NXdetector/polar\_angle} of length \texttt{NP} containing the positions of the detector's polar\_angle at the various steps of the scan.
- A dataset at \texttt{NXentry/NXsample/rotation\_angle} of length \texttt{NP} containing the positions of \texttt{rotation\_angle} at the various steps of the scan.
- A dataset at \texttt{NXentry/NXsample/chi} of length \texttt{NP} containing the positions of \texttt{chi} at the various steps of the scan.
- A dataset at \texttt{NXentry/NXsample/phi} of length \texttt{NP} containing the positions of \texttt{phi} at the various steps of the scan.
- A dataset at \texttt{NXentry/NXsample/h} of length \texttt{NP} containing the positions of the reciprocal coordinate \texttt{h} at the various steps of the scan.
- A dataset at \texttt{NXentry/NXsample/k} of length \texttt{NP} containing the positions of the reciprocal coordinate \texttt{k} at the various steps of the scan.
- A dataset at \texttt{NXentry/NXsample/l} of length \texttt{NP} containing the positions of the reciprocal coordinate \texttt{l} at the various steps of the scan.
- \texttt{NXdata} contains links to:
  - \texttt{NXentry/NXinstrument/NXdetector/data}
  - \texttt{NXentry/NXinstrument/NXdetector/polar\_angle}
  - \texttt{NXentry/NXsample/rotation\_angle}
  - \texttt{NXentry/NXsample/chi}
  - \texttt{NXentry/NXsample/phi}
  - \texttt{NXentry/NXsample/h}
  - \texttt{NXentry/NXsample/k}
  - \texttt{NXentry/NXsample/l}

The \texttt{NXdata} also contains appropriate attributes as described in \textit{Associating plottable data using attributes applied to the NXdata group}.

- All other fields have their normal dimensions.

\textbf{NeXus Complex \textit{hkl} Scan}

```python
entry:NXentry
  @default = data
  instrument:NXinstrument
detector:NXdetector
  data[NP]
polar\_angle[NP]
  name
  sample:NXsample
```

(continues on next page)
Multi-parameter scan: XAS

Data can be stored almost anywhere in the NeXus tree. While the previous examples showed data arrays in either NXdetector or NXsample, this example demonstrates that data can be stored in other places. Links are used to reference the data.

The example is for X-ray Absorption Spectroscopy (XAS) data where the monochromator energy is step-scanned and counts are read back from detectors before ($I_0$) and after ($I$) the sample. These energy scans are repeated at a sequence of sample temperatures to map out, for example, a phase transition. While it is customary in XAS to plot $\log(I_0/I)$, we show them separately here in two different NXdata groups to demonstrate that such things are possible. Note that the length of the 1-D energy array is $NE$ while the length of the 1-D temperature array is $NT$.

NeXus Multi-parameter scan: XAS

```plaintext
entry:NXentry
    @default = "I_data"
    instrument:NXinstrument
        I:NXdetector
            data:NX_NUMBER[NE,NT]
            energy --> /entry/monochromator/energy
            temperature --> /entry/sample/temperature
            I0:NXdetector
```
Rastering

Rastering is the process of making experiments at various locations in the sample volume. Again, rasterisation experiments can be variable. Some people even raster on spirals! Rasterisation experiments are treated the same way as described above for scans. Just replace \( NP \) with \( P \), the number of raster points.

Special rules apply if a rasterisation happens on a regular grid of size \( xraster, yraster \). Then the variables varied in the rasterisation will be of dimensions \( xraster, yraster \) and the detector data of dimensions \( xraster, yraster, (original dimensions) \) of the detector. For example, an area detector of size \( xsize, ysize \) then it is stored with dimensions \( xraster, yraster, xsize, ysize \).

**Warning:** Be warned: if you use the 2D rasterisation method with \( xraster, yraster \) you may end up with invalid data if the scan is aborted prematurely. This cannot happen if the first method is used.

Streaming Data Acquisition And Logging

More and more data is collected in streaming mode. This means that time stamped data is logged for one or more inputs, possibly together with detector data. Another use case is the logging of parameters, for example temperature, while a long running data collection is in progress. NeXus covers this case too. There is one simple rule for structuring such files:

Just use the standard NeXus raw data file structure, but replace the corresponding data object with an \( NXlog \) or \( NX-event_data \) structure of the same name.

For example, consider your instrument is streaming detector images against a magnetic_field on the sample. In this case both NXsample/magnetic_field and NXdetector/data would become NXlog structures instead of simple arrays i.e.

```python
# data
data: NX_NUMBER[NE,NT]
enery --> /entry/monochromator/energy
temperature --> /entry/sample/temperature

# sample
sample: NXsample
temperature: NX_NUMBER[NT]
monochromator: NXmonochromator
energy: NX_NUMBER[NE]

# I0_data
I0_data: NXdata
@signal = "data"
@axes = ["energy", "temperature"]
@energy_indices = 0
@temperature_indices = 0
data --> /entry/instrument/I00/data
energy --> /entry/monochromator/energy
temperature --> /entry/sample/temperature
```

```python
# I_data
I_data: NXdata
@signal = "data"
@axes = ["energy", "temperature"]
@energy_indices = 0
@temperature_indices = 0
data --> /entry/instrument/I/data
energy --> /entry/monochromator/energy
temperature --> /entry/sample/temperature
```
the NXlog structure will have the same name as the NeXus field involved.

**NXcollection**

On demand from the community, NeXus introduced a more informal method of storing information in a NeXus file. This is the `NXcollection` class which can appear anywhere underneath `NXentry`. `NXcollection` is a container for holding other data. The foreseen use is to document collections of similar data which do not otherwise fit easily into the `NXinstrument` or `NXsample` hierarchy, such as the intent to record all motor positions on a synchrotron beamline. Thus, `NXcollection` serves as a quick point of access to data for an instrument scientist or another expert. `NXcollection` is also a feature for those who are too lazy to build up the complete NeXus hierarchy. An example usage case is documented in figure `NXcollection example`.

**NXcollection Example**

```
entry:NXentry
    positioners:NXcollection
        mxx:NXpositioner
        mzz:NXpositioner
        squ:NXpositioner
        ttv:NXpositioner
        hugo:NXpositioner
        ....
    scalars:NXcollection
        title (dataset)
        lieselotte (dataset)
        ...
    detectors:NXcollection
        Pilatus:NXdata
        MXX-45:NXdata
        ....
```

**Rules for Storing Data Items in NeXus Files**

This section describes the rules which apply for storing single data items.

**Naming Conventions**

Group and field names used within NeXus follow a naming convention described by the following rules:

- The names of NeXus group and field items must only contain a restricted set of characters. This set is described by a regular expression syntax regular expression syntax, as described below.

- For the class names of NeXus group items, the prefix `NX` is reserved as shown in the table below. Thus all NeXus class names start with NX. The chapter titled *NeXus: Reference Documentation* lists the available NeXus class names as either base classes, application definitions, or contributed definitions.

---

1 The class name is the value assigned to the `NX_class` attribute of an HDF5 group in the NeXus data file. This class name is different than the name of the HDF5 group. This is important when not using the NAPI to either read or write the HDF5 data file.
NXDL group and field names

The names of NeXus group and field items are validated according to these boundaries:

- **Recommended** names:
  - lower case words separated by underscores and, if needed, with a trailing number
  - NOTE: this is used by the NeXus base classes

- **Allowed** names:
  - any combination of upper and lower case letter, numbers, underscores and periods, except that periods cannot be at the start or end of the string
  - NOTE: this matches the `validItemName` regular expression below

- **Invalid** names:
  - NOTE: does not match the `validItemName` regular expression below

Regular expression pattern for NXDL group and field names

The NIAC recognises that the majority of the world uses characters outside of the basic latin (a.k.a. US-ASCII, 7-bit ASCII) set currently included in the allowed names. The restriction given here reflects current technical issues and we expect to revisit the issue and relax such restrictions in future.

The names of NeXus group and field items must match this regular expression (named `validItemName` in the XML Schema file: `nxdl.xsd`):

```
^[a-zA-Z0-9_]([a-zA-Z0-9_.]*[a-zA-Z0-9_.])?$
```

The length should be limited to no more than 63 characters (imposed by the HDF5 rules for names).

It is recognized that some facilities will construct data files with group and field names with upper case letters or start names with a number or include a period in a name.

Use of underscore in descriptive names

Sometimes it is necessary to combine words in order to build a descriptive name for a field or a group. In such cases lowercase words are connected by underscores.

```
number_of_lenses
```

For all fields, only names from the NeXus base class dictionaries should be used. If a field name or even a complete component is missing, please suggest the addition to the **NIAC: The NeXus International Advisory Committee**. The addition will usually be accepted provided it is not a duplication of an existing field and adequately documented.

**Note:** The NeXus base classes provide a comprehensive dictionary of terms that can be used for each class. The expected spelling and definition of each term is specified in the base classes. It is not required to provide all the terms specified in a base class. Terms with other names are permitted but might not be recognized by standard software. Rather than persist in using names not specified in the standard, please suggest additions to the **NIAC: The NeXus International Advisory Committee**.

---

3 NeXus data files with group or field names that match the regular expression but contain upper case characters, start with a digit, or include a period in the group or field names might not be accepted by all software that reads NeXus data files. These names will be flagged as a warning during data file validation.
The data stored in NeXus fields must be readback values. This means values as read from the detector, other hardware, etc. There are occasions where it is sensible to store the target value the variable was supposed to have. In such cases, the target value is stored with a name built by appending _set to the NeXus (readback) field name.

Consider this example:

```
1 temperature
2 temperature_set
```

The `temperature` field will hold the readback from the cryostat/furnace/whatever. The field `temperature_set` will hold the target value for the temperature as set by the experiment control software.

Some fields share a common part of their name and an additional part name that makes the whole name specific. For example, a `unit_cell` might have parts named abc, alphabetagamma, and volume. It is recommended to write them with the common part first, an underscore (\_), and then the specific part. In this way, the fields will sort alphabetically on the common name. So, in this example:

```
1 unit_cell_abc
2 unit_cell_alphabetagamma
3 unit_cell_volume
```

### Reserved prefixes

When naming an attribute, field, or group, NeXus has reserved certain prefixes to the names to ensure that names written in NeXus files will not conflict with future releases as the NeXus standard evolves. Prefixes should follow a naming scheme of uppercase letters followed by an underscore, but exceptions will be made for cases already in wide use. The following table lists the prefixes reserved by NeXus.

<table>
<thead>
<tr>
<th>prefix</th>
<th>use</th>
<th>meaning</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLUESKY_</td>
<td>attributes</td>
<td>reserved for use by Bluesky project</td>
<td><a href="https://blueskyproject.io">https://blueskyproject.io</a></td>
</tr>
<tr>
<td>DECTRIS_</td>
<td>attributes, fields</td>
<td>reserved for use by Dectris</td>
<td><a href="https://www.dectris.com">https://www.dectris.com</a></td>
</tr>
<tr>
<td>IDF_</td>
<td>attributes</td>
<td>reserved for use by pulsedTD Muon definition</td>
<td><a href="https://www.isis.stfc.ac.uk/Pages/nexus-definition-v27924.pdf">https://www.isis.stfc.ac.uk/Pages/nexus-definition-v27924.pdf</a></td>
</tr>
<tr>
<td>NDAttr</td>
<td>attributes</td>
<td>reserved for use by EPICS area detector</td>
<td><a href="https://github.com/areaDetector">https://github.com/areaDetector</a></td>
</tr>
<tr>
<td>NX</td>
<td>NXDL class</td>
<td>for the class names used with NeXus groups</td>
<td><a href="https://www.nexusformat.org">https://www.nexusformat.org</a></td>
</tr>
<tr>
<td>NX_</td>
<td>attributes</td>
<td>reserved for use by NeXus</td>
<td><a href="https://www.nexusformat.org">https://www.nexusformat.org</a></td>
</tr>
<tr>
<td>PDBX_</td>
<td>attributes</td>
<td>reserved for the US protein data bank</td>
<td><a href="https://www.rcsb.org">https://www.rcsb.org</a></td>
</tr>
<tr>
<td>SAS_</td>
<td>attributes</td>
<td>reserved for use by canSAS</td>
<td><a href="https://www.cansas.org">https://www.cansas.org</a></td>
</tr>
<tr>
<td>SILX_</td>
<td>attributes</td>
<td>reserved for use by silx</td>
<td><a href="https://www.silx.org">https://www.silx.org</a></td>
</tr>
</tbody>
</table>
Reserved suffixes

When naming a field, NeXus has reserved certain suffixes to the names so that a specific meaning may be attached. Consider a field named \texttt{DATASET}, the following table lists the suffixes reserved by NeXus.

<table>
<thead>
<tr>
<th>suffix</th>
<th>reference</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>_end</td>
<td>\texttt{NXtransformations}</td>
<td>end points of the motions that start with \texttt{DATASET}</td>
</tr>
<tr>
<td>_errors</td>
<td>\texttt{NXdata}</td>
<td>uncertainties (a.k.a., errors)</td>
</tr>
<tr>
<td>_increment</td>
<td>\texttt{NXtransformations}</td>
<td>intended average range through which the corresponding axis moves during the exposure of a frame</td>
</tr>
<tr>
<td>_indices</td>
<td>\texttt{NXdata}</td>
<td>Integer array that defines the indices of the signal field which need to be used in the \texttt{DATASET} in order to reference the corresponding axis value</td>
</tr>
<tr>
<td>_mask</td>
<td></td>
<td>Field containing a signal mask, where 0 means the pixel is not masked. If required, bit masks are defined in \texttt{NXdetector pixel_mask}.</td>
</tr>
<tr>
<td>_set</td>
<td>\texttt{target values}</td>
<td>Target value of \texttt{DATASET}</td>
</tr>
<tr>
<td>_weights</td>
<td></td>
<td>divide \texttt{DATASET} by these weights</td>
</tr>
</tbody>
</table>

Variants

Sometimes it is necessary to store alternate values of a NeXus field in a NeXus file. A common example may be the beam center of which a rough value is available at data acquisition. But later on, a better beam center is calculated as part of the data reduction. In order to store this without losing the historical information, the original field can be given a variant attribute that points to a new field containing the obsolete value. If even better values become available, further fields can be inserted into the chain of variant attributes pointing to the preceeding value for the field. A reader can thus keep the best value in the pre-defined field, and also be able to follow the variant chain and locate older variants.

A little example is in order to illustrate the scheme:

```plaintext
beam_center_x
variant=beam_center_x_refined
beam_center_x_refined
variant=beam_center_x_initial_guess
beam_center_x_initial_guess
```

NeXus borrowed this scheme from CIF. In this way all the different variants of a field can be preserved. The expectation is that variants will be rarely used and NXprocess groups with the results of data reduction will be written instead.

Uncertainties or Errors

It is desirable to store experimental errors (also known as uncertainties) together with the data. NeXus supports this through a convention: uncertainties or experimental errors on data are stored in a separate field which has a name consisting of the original name of the data with \_\texttt{errors} appended to it. These uncertainties fields have the same shape as the original data field.

An example, from \texttt{NXdetector}:

\footnote{If \texttt{DATASET_weights} exists and has the same shape as the field, you are supposed to divide \texttt{DATASET} by the weights.}
Where data errors would contain the errors on data, and beam_center_x_errors the error on the beam center for x.

NeXus Array Storage Order

NeXus stores multi-dimensional arrays of physical values in C language storage order, where the first dimension has the slowest varying index when iterating through the array in storage order, and the last dimension is the fastest varying. This is the rule. Good reasons are required to deviate from this rule.

Where the array contains data from a detector, the array dimensions may correspond to physical directions or axes. The slowest, slow, fast, fastest qualifiers can then apply to these axes too.

It is possible to store data in storage orders other than C language order.

As well it is possible to specify that the data needs to be converted first before being useful. Consider one situation, when data must be streamed to disk as fast as possible and conversion to C language storage order causes unnecessary latency. This case presents a good reason to make an exception to the standard rule.

Non C Storage Order

In order to indicate that the storage order is different from C storage order two additional data set attributes, offset and stride, have to be stored which together define the storage layout of the data. Offset and stride contain rank numbers according to the rank of the multidimensional data set. Offset describes the step to make when the dimension is multiplied by 1. Stride defines the step to make when incrementing the dimension. This is best explained by some examples.

Offset and Stride for 1 D data:

```plaintext
* raw data = 0 1 2 3 4 5 6 7 8 9
  size[1] = { 10 } // assume uniform overall array dimensions

* default stride:
  stride[1] = { 1 }
  offset[1] = { 0 }
  for i:
      result[i]:
          0 1 2 3 4 5 6 7 8 9

* reverse stride:
  stride[1] = { -1 }
  offset[1] = { 9 }
  for i:
      result[i]:
          9 8 7 6 5 4 3 2 1 0
```
Offset and Stride for 2D Data

```
/* raw data = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
   size[2] = { 4, 5 } // assume uniform overall array dimensions

/* row major (C) stride:
   stride[2] = { 5, 1 }
   offset[2] = { 0, 0 }
   for i:
      for j:
         result[i][j]:
            0 1 2 3 4
            5 6 7 8 9
            10 11 12 13 14
            15 16 17 18 19

/* column major (Fortran) stride:
   stride[2] = { 1, 4 }
   offset[2] = { 0, 0 }
   for i:
      for j:
         result[i][j]:
            0 1 2 3 4
            5 6 7 8 9
            10 11 12 13 14
            15 16 17 18 19

/* "crazy reverse" row major (C) stride:
   stride[2] = { -5, -1 }
   offset[2] = { 4, 5 }
   for i:
      for j:
         result[i][j]:
            9 8 7 6 5
            4 3 2 1 0
```

Offset and Stride for 3D Data

```
/* raw data = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
   20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
   40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
   size[3] = { 3, 4, 5 } // assume uniform overall array dimensions

/* row major (C) stride:
   stride[3] = { 20, 5, 1 }
   offset[3] = { 0, 0, 0 }
   for i:
      for j:
         for k:
```

(continues on next page)
result[i][j][k]:
  0  1  2  3  4
  5  6  7  8  9
 10 11 12 13 14
 15 16 17 18 19

 20 21 22 23 24
 25 26 27 28 29
 30 31 32 33 34
 35 36 37 38 39

 40 41 42 43 44
 45 46 47 48 49
 50 51 52 53 54
 55 56 57 58 59

* column major (Fortran) stride:
  stride[3] = { 1, 3, 12 }
  offset[3] = { 0, 0, 0 }
for i:
  for j:
    for k:
      result[i][j][k]:
        0 12 24 36 48
        3 15 27 39 51
        6 18 30 42 54
        9 21 33 45 57

        1 13 25 37 49
        4 16 28 40 52
        7 19 31 43 55
        10 22 34 46 58

        2 14 26 38 50
        5 17 29 41 53
        8 20 32 44 56
        11 23 35 47 59

**NeXus Data Types**

<table>
<thead>
<tr>
<th>description</th>
<th>matching regular expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>NX_INT(8</td>
</tr>
<tr>
<td>floating-point</td>
<td>NX_FLOAT(32</td>
</tr>
<tr>
<td>array</td>
<td>([0-9])?</td>
</tr>
<tr>
<td>valid item name</td>
<td>^[a-zA-Z0-9_.]<em>[a-zA-Z0-9_.]</em>$</td>
</tr>
<tr>
<td>valid class name</td>
<td>^NX[A-Za-z0-9_]*$</td>
</tr>
</tbody>
</table>

NeXus supports numeric data as either integer or floating-point numbers. A number follows that indicates the number of bits in the word. The table above shows the regular expressions that match the data type specifier.
integers
   NX_INT8, NX_INT16, NX_INT32, or NX_INT64

floating-point numbers
   NX_FLOAT32 or NX_FLOAT64

date / time stamps
   NX_DATE_TIME or ISO8601: Dates and times are specified using ISO-8601 standard definitions. Refer to NeXus dates and times.

strings
   NX_CHAR: The preferred string representation is UTF-8. Both fixed-length strings and variable-length strings are valid. String arrays cannot be used where only a string is expected (title, start_time, end_time, NX_class attribute,...). Fields or attributes requiring the use of string arrays will be clearly marked as such (like the NXdata attribute auxiliary_signals).

binary data
   Binary data is to be written as UINT8.

images
   Binary image data is to be written using UINT8, the same as binary data, but with an accompanying image mime-type. If the data is text, the line terminator is [CR][LF].

NeXus dates and times

NeXus dates and times should be stored using the ISO 8601\(^5\) format, e.g. 1996-07-31T21:15:22+0600 (which includes a time zone offset of +0600). Note: The time zone offset is always numeric or Z (which means UTC). The standard also allows for time intervals in fractional seconds with 1 or more digits of precision. This avoids confusion, e.g. between U.S. and European conventions, and is appropriate for machine sorting. It is recommended to add an explicit time zone, otherwise the local time zone is assumed per ISO8601. The norm is that if there is no time zone, it is assumed local time, however, when a file moves from one country to another it is undefined. If the local time zone is written, the ambiguity is gone.

strftime() format specifiers for ISO-8601 time

%Y-%m-%dT%H:%M:%S%z

Note: Note that the T appears literally in the string, to indicate the beginning of the time element, as specified in ISO 8601. It is common to use a space in place of the T, such as 1996-07-31 21:15:22+0600. While human-readable (and later allowed in a relaxed revision of the standard), compatibility with libraries supporting the ISO 8601 standard is not assured with this substitution. The strftime() format specifier for this is “%Y-%m-%d %H:%M:%S%z”.

\(^5\) ISO 8601: https://www.w3.org/TR/NOTE-datetime
NeXus Data Units

Given the plethora of possible applications of NeXus, it is difficult to define units to use. Therefore, the general rule is that you are free to store data in any unit you find fit. However, any field must have a units attribute which describes the units. Wherever possible, SI units are preferred. NeXus units are written as a string attribute (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. Application definitions may specify units to be used for fields using an enumeration.

Storing Detectors

There are very different types of detectors out there. Storing their data can be a challenge. As a general guide line: if the detector has some well defined form, this should be reflected in the data file. A linear detector becomes a linear array, a rectangular detector becomes an array of size xsize times ysize. Some detectors are so irregular that this does not work. Then the detector data is stored as a linear array, with the index being detector number till ndet. Such detectors must be accompanied by further arrays of length ndet which give azimuthal_angle, polar_angle and distance for each detector.

If data from a time of flight (TOF) instrument must be described, then the TOF dimension becomes the last dimension, for example an area detector of xsize vs. ysize is stored with TOF as an array with dimensions xsize, ysize, ntof.

Monitors are Special

Monitors, detectors that measure the properties of the experimental probe rather than the probe’s interaction with the sample, have a special place in NeXus files. Monitors are crucial to normalize data. To emphasize their role, monitors are not stored in the NXinstrument hierarchy but on NXentry level in their own groups as there might be multiple monitors. Of special importance is the monitor in a group called control. This is the main monitor against which the data has to be normalized. This group also contains the counting control information, i.e. counting mode, times, etc.

Monitor data may be multidimensional. Good examples are scan monitors where a monitor value per scan point is expected or time-of-flight monitors.

Find the plottable data

Simple plotting is one of the motivations for the NeXus standard. To implement simple plotting, a mechanism must exist to identify the default data for visualization (plotting) in any NeXus data file. Over its history the NIAC has agreed upon a method of applying metadata to identify the default plottable data. This metadata has always been specified as HDF attributes. With the evolution of the underlying file formats and the NeXus data standard, the method to identify the default plottable data has evolved, undergoing three distinct versions.

version 1
Associating plottable data by dimension number using the axis attribute

version 2
Associating plottable data by name using the axes attribute

version 3
Associating plottable data using attributes applied to the NXdata group

The UDunits specification also includes instructions for derived units. At present, the contents of NeXus units attributes are not validated in data files.
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.

For programmers who may encounter NeXus data files written using any of these methods, we present the algorithm for each method to find the default plottable data. It is recommended to start with the most recent method, *Version 3*, first.

**Version 3**

The third (current) method to identify the default plottable data is as follows:

1. Start at the top level of the NeXus data file (the *root* of the HDF5 hierarchy).
2. Pick the default *NXentry* group.

   If the *root* has an attribute `default`, the attribute’s value is the name of the *NXentry* group to be used. (The value of the `default` attribute names an existing child of this group. The child group must itself be a NeXus group.) If no `default` attribute exists, pick any *NXentry* group. This is trivial if there is only one *NXentry* group.

![Diagram of NXentry selection](image)

**Fig. 6: Find plottable data: select the NXentry group**

3. Pick the default *NXdata* group.

   Open the *NXentry* group selected above. If it has an attribute `default`, the attribute’s value is the name of the *NXdata* group to be used. (The value of the `default` attribute names an existing child of this group. The child group must itself be a NeXus group.) If no `default` attribute exists, pick any *NXdata* group. This is trivial if there is only one *NXdata* group.

![Diagram of NXdata selection](image)

**Fig. 7: Find plottable data: select the NXdata group**

1. Pick the default plottable field (the *signal* data).

   Open the *NXdata* group selected above. If it has a *signal* attribute, the attribute’s value is the name of the field to be plotted. (The value of the *signal* attribute names an existing child of this group. The child group must itself be a NeXus field.) If no *signal* attribute is present on the *NXdata* group, then proceed to try an *older NeXus method* to find the default plottable data.
Fig. 8: Find plottable data: select the signal data

1. Pick the fields with the dimension scales (the axes).

If the same NXdata group has an attribute axes, then its value is a string (signal data is 1-D) or string array (signal data is 2-D or higher rank) naming the field in this group to be used as dimension scales of the default plottable data. The number of values given must be equal to the rank of the signal data. These are the abscissae of the plottable signal data.

If no field is available to provide a dimension scale for a given dimension, then a “.” will be used in that position. In such cases, programmers are expected to use an integer sequence starting from 0 for each position along that dimension.

2. Associate the dimension scales with each dimension of the plottable data.

For each field (its name is AXISNAME) in axes that provides a dimension scale, there will be an NXdata group attribute AXISNAME_indices which value is an .. integer or integer array with value of the dimensions of the signal data to which this dimension scale applies.

If no AXISNAME_indices attribute is provided, a programmer is encouraged to make best efforts assuming the intent of this NXdata group to provide a default plot. The AXISNAME_indices attribute is only required when necessary to resolve ambiguity.

It is possible there may be more than one AXISNAME_indices attribute with the same value or values. This indicates the possibility of using alternate abscissae along this (these) dimension(s). The field named in the axes attribute indicates the intention of the data file writer as to which field should be used by default.

2. Plot the signal data, given axes and AXISNAME_indices.

When all the default and signal attributes are present, this Python code example will identify directly the default plottable data (assuming a plot() function has been defined by some code:

```python
import h5py

with h5py.File(hdf5_file_name, "r") as group:
    while "default" in group.attrs:
        child_group_name = group.attrs["default"]
        group = group[child_group_name]

    if "NX_class" in group.attrs:
        signal_field_name = group.attrs["signal"]
        data = group[signal_field_name]

    plot(data)
```
Version 2

Tip: Try this method for older NeXus data files and Version 3 fails.

The second method to identify the default plottable data is as follows:

1. Start at the top level of the NeXus data file.
2. Loop through the groups with class NXentry until the next step succeeds.

3. Open the NXentry group and loop through the subgroups with class NXdata until the next step succeeds.

4. Open the NXdata group and loop through the fields for the one field with attribute signal="1". Note: There should be only one field that matches.

   1. If this field has an attribute axes:
      1. The axes attribute value contains a colon (or comma) delimited list (in the C-order of the data array) with the names of the dimension scales associated with the plottable data. Such as: axes="polar_angle:time_of_flight"
      2. Parse axes and open the fields to describe your dimension scales
   2. If this field has no attribute axes:
      1. Search for fields with attributes axis=1, axis=2, etc.
      2. These are the fields describing your axis. There may be several fields for any axis, i.e. there may be multiple fields with the attribute axis=1. Among them the field with the attribute primary=1 is the preferred one. All others are alternative dimension scales.

5. Having found the default plottable data and its dimension scales: make the plot.
Version 1

**Tip:** Try this method for older NeXus data files.

The first method to identify the default plottable data is as follows:

1. Open the first top level NeXus group with class `NXentry`.

   ![Diagram](image1)

   Fig. 12: Find plottable data: pick the first `NXentry` group

2. Open the first NeXus group with class `NXdata`.

   ![Diagram](image2)

   Fig. 13: Find plottable data: pick the first `NXdata` group

3. Loop through NeXus fields in this group searching for the item with attribute `signal="1"` indicating this field has the plottable data.

4. Search for the one-dimensional NeXus fields with attribute `primary=1`. These are the dimension scales to label the axes of each dimension of the data.

5. Link each dimension scale to the respective data dimension by the `axis` attribute (`axis=1`, `axis=2`, ... up to the rank of the data).

6. If necessary, close this `NXdata` group, search the next `NXdata` group, repeating steps 3 to 5.

7. If necessary, close the `NXentry` group, search the next `NXentry` group, repeating steps 2 to 6.
Fig. 14: Find plottable data: select the signal data

**Associating Multi Dimensional Data with Axis Data**

NeXus allows for storage of multi dimensional arrays of data. It is this data that presents the most challenge for description. In most cases it is not sufficient to just have the indices into the array as a label for the dimensions of the data. Usually the information which physical value corresponds to an index into a dimension of the multi dimensional data set. To this purpose a means is needed to locate appropriate data arrays which describe what each dimension of a multi dimensional data set actually corresponds too. There is a standard HDF facility to do this: it is called dimension scales. Unfortunately, when NeXus was first designed, there was only one global namespace for dimension scales. Thus NeXus had to devise its own scheme for locating axis data which is described here. A side effect of the NeXus scheme is that it is possible to have multiple mappings of a given dimension to physical data. For example, a TOF data set can have the TOF dimension as raw TOF or as energy.

There are now three methods of associating each data dimension to its respective dimension scale. Only the first method is recommended now, the other two (older methods) are now discouraged.

1. **Associating plottable data using attributes applied to the NXdata group**
2. **Associating plottable data by name using the axes attribute**
3. **Associating plottable data by dimension number using the axis attribute**

The recommended method uses the axes attribute applied to the NXdata group to specify the names of each dimension scale. A prerequisite is that the fields describing the axes of the plottable data are stored together with the plottable data in the same NeXus group. If this leads to data duplication, use links.

---

**Associating plottable data using attributes applied to the NXdata group**

**Tip:** Recommended: This is the “NIAC2014” method recommended for all new NeXus data files.

The default data to be plotted (and any associated axes) is specified using attributes attached to the NXdata group.

**signal**

Defines the name of the default field in the NXdata group. A field of this name must exist (either as field or link to field).

It is recommended to use this attribute rather than adding a signal attribute to the field. The procedure

7 Summary of the discussion at NIAC2014 to revise how to find default data: https://www.nexusformat.org/2014_How_to_find_default_data.html
to identify the default data to be plotted is quite simple. Given any NeXus data file, any NXentry, or any NXdata, follow the chain as it is described from that point. Specifically:

- The root of the NeXus file may have a default attribute that names the default NXentry group. This attribute may be omitted if there is only one NXentry group. If a second NXentry group is later added, the default attribute must be added then.

- Every NXentry group may have a default attribute that names the default NXdata group. This attribute may be omitted if there is only one NXdata group or if no NXdata is present. If a second NXdata group is later added, the default attribute must be added then.

- Every NXdata group will have a signal attribute that names the field name to be plotted by default. This attribute is required.

**axes**

String array\(^8\) that defines the independent data fields used in the default plot for all of the dimensions of the signal field. One entry is provided for every dimension in the signal field.

The field(s) named as values (known as “axes”) of this attribute must exist. An axis slice is specified using a field named AXISNAME_indices as described below (where the text shown here as AXISNAME is to be replaced by the actual field name).

When no default axis is available for a particular dimension of the plottable data, use a “.” in that position.

See examples provided on the NeXus webpage \(^9\).

If there are no axes at all (such as with a stack of images), the axes attribute can be omitted.

**AXISNAME_indices**

Each AXISNAME_indices attribute indicates the dependency relationship of the AXISNAME field (where AXISNAME is the name of a field that exists in this NXdata group) with one or more dimensions of the plottable data.

Integer array\(^8\) that defines the indices of the signal field (that field will be a multidimensional array) which need to be used in the AXISNAME field in order to reference the corresponding axis value.

The first index of an array is 0 (zero).

Here, AXISNAME is to be replaced by the name of each field described in the axes attribute. An example with 2-D data, \(d(t, P)\), will illustrate:

```plaintext
data_2d:NXdata
   @signal="data"
   @axes=["time","pressure"]
   @time_indices=0
   @pressure_indices=1
   data: float[1000,20]
   time: float[1000]
   pressure: float[20]
```

This attribute is to be provided in all situations. However, if the indices attributes are missing (such as for data files written before this specification), file readers are encouraged to make their best efforts to plot the data. Thus the implementation of the AXISNAME_indices attribute is based on the model of “strict writer, liberal reader”.

\(^8\) Note on array attributes: Attributes potentially containing multiple values (axes and _indices) are to be written as string or integer arrays, to avoid string parsing in reading applications.

\(^9\) NIAC2014 proposition: https://www.nexusformat.org/2014_axes_and_uncertainties.html
Examples

Several examples are provided to illustrate this method. More examples are available in the NeXus webpage (\(^9\)).

**simple 1-D data example showing how to identify the default data (counts vs. mr)**

In the first example, storage of a 1-D data set (counts vs. mr) is described.

```
datafile.hdf5:NeXus data file
  @default="entry"
  entry:NXentry
    @default="data"
    data:NXdata
      @signal="counts"
      @axes="mr"
      @mr_indices=0
      counts: float[100] --> the default dependent data
    mr: float[100] --> the default independent data
```

**2-D data example showing how to identify the default data and associated dimension scales**

A 2-D data set, data as a function of time and pressure is described. By default as indicated by the axes attribute, pressure is to be used. The temperature array is described as a substitute for pressure (so it replaces dimension 1 of data as indicated by the temperature_indices attribute).

```
datafile.hdf5:NeXus data file
  @default="entry"
  entry:NXentry
    @default="data_2d"
    data_2d:NXdata
      @signal="data"
      @axes=['"time","pressure"]
      @pressure_indices=1
      @temperature_indices=1
      @time_indices=0
      data: float[1000,20]
      pressure: float[20]
      temperature: float[20]
      time: float[1000]
```

**Associating plottable data by name using the axes attribute**

**Warning:** Discouraged: See this method: *Associating plottable data using attributes applied to the NXdata group.*

This method defines an attribute of the data field called axes. The axes attribute contains the names of each dimension scale as a colon (or comma) separated list in the order they appear in C. For example:
### denoting axes by name

```plaintext
data:NXdata
time_of_flight = 1500.0 1502.0 1504.0 ...
polar_angle = 15.0 15.6 16.2 ...
some_other_angle = 0.0 0.0 2.0 ...
data = 5 7 14 ...
@axes = ["polar_angle", "time_of_flight"]
@signal = 1
```

### Associating plottable data by dimension number using the `axis` attribute

**Warning:** Discouraged: See this method: *Associating plottable data by name using the `axes` attribute*

The original method defines an attribute of each dimension scale field called `axis`. It is an integer whose value is the number of the dimension, in order of fastest varying dimension. That is, if the array being stored is data with elements `data[i][j]` 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:

### denoting axes by integer number

```plaintext
data:NXdata
time_of_flight = 1500.0 1502.0 1504.0 ...
    @axis = 1
    @primary = 1
polar_angle = 15.0 15.6 16.2 ...
    @axis = 2
    @primary = 1
some_other_angle = 0.0 0.0 2.0 ...
    @axis = 1
    @primary = 1
data = 5 7 14 ...
    @signal = 1
```

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

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 the dimension scales 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.

---

**Note:**
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.

Physical File format

This section describes how NeXus structures are mapped to features of the underlying physical file format. This is a guide for people who wish to create NeXus files without using the NeXus-API.

Choice of HDF as Underlying File Format

At its beginnings, the founders of NeXus identified the Hierarchical Data Format (HDF) as a capable and efficient multi-platform data storage format. HDF was designed for large data sets and already had a substantial user community. HDF was developed and maintained initially by 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: http://www.hdfgroup.org/). Rather then developing its own unique physical file format, the NeXus group choose to build NeXus on top of HDF.

HDF (now HDF5) is 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.

Mapping NeXus into HDF

NeXus data structures map directly to HDF structures. NeXus groups are HDF5 groups and NeXus fields (or data sets) are HDF5 datasets. Attributes map directly to HDF group or dataset attributes. The NeXus class is stored as an attribute to the HDF5 group with the name NX_class with value of the NeXus class name. (For legacy NeXus data files using HDF4, groups are HDF4 vgroups and fields are HDF4 SDS (scientific data sets). HDF4 does not support group attributes. HDF4 supports a group class which is set with the Vsetclass() call and read with VGetclass().)

A NeXus link directly maps to the HDF hard link mechanisms.

Note: Examples are provided in the Examples of writing and reading NeXus data files chapter. These examples include software to write and read NeXus data files using the NAPI, as well as other software examples that use native (non-NAPI) libraries. In some cases the examples show the content of the NeXus data files that are produced. Here are links to some of the examples:

- How do I write a NeXus file?
- How do I read a NeXus file?
- HDF5 in C with NAPI
  - HDF5 in Python with NAPI
  - Writing a simple NeXus file using native HDF5 commands in C
  - Reading a simple NeXus file using native HDF5 commands in C
  - Write a NeXus HDF5 File
  - Read a NeXus HDF5 File
Perhaps the easiest way to view the implementation of NeXus in HDF5 is to look at the data structure. For this, we use the `h5dump` command-line utility provided with the HDF5 support libraries. Short examples are provided for the basic NeXus data components:

- **group**: created in C NAPI by:
  ```
  NXmakegroup (fileID, "entry", "NXentry");
  ```

- **field**: created in C NAPI by:
  ```
  NXmakedata (fileID, "two_theta", NX_FLOAT32, 1, &n);
  NXopendata (fileID, "two_theta");
  NXputdata (fileID, tth);
  ```

- **attribute**: created in C NAPI by:
  ```
  NXputattr (fileID, "units", "degrees", 7, NX_CHAR);
  ```

- **link**: created in C NAPI by:
  ```
  NXmakelink (fileid, &itemid);
  # -or-
  NXmakenamedlink (fileid, "linked_name", &itemid);
  ```

### h5dump of a NeXus NXentry group

```
GROUP "entry" {
  ATTRIBUTE "NX_class" {
    DATATYPE H5T_STRING {
      STRSIZE 7;
      STRPAD H5T_STR_NULLPAD;
      CSET H5T_CSET_ASCII;
      CTYPE H5T_C_S1;
    }
    DATASPACE SCALAR
    DATA {
      (0): "NXentry"
    }
  }
  # ... group contents
}
```
h5dump of a NeXus field (HDF5 dataset)

```hdf5
DATASET "two_theta" {
    DATATYPE H5T_IEEE_F64LE
    DATASPACE SIMPLE { ( 31 ) / ( 31 ) }
    DATA {
        (0): 17.9261, 17.9259, 17.9258, 17.9256, 17.9254, 17.9252,
        (6): 17.9251, 17.9249, 17.9247, 17.9246, 17.9244, 17.9243,
        (12): 17.9241, 17.9239, 17.9237, 17.9236, 17.9234, 17.9232,
        (18): 17.9231, 17.9229, 17.9228, 17.9226, 17.9224, 17.9222,
        (24): 17.9221, 17.9219, 17.9217, 17.9216, 17.9214, 17.9213,
        (30): 17.9211
    }
    ATTRIBUTE "units" {
        DATATYPE H5T_STRING {
            STRSIZE 7;
            STRPAD H5T_STR_NULLPAD;
            CSET H5T_CSET_ASCII;
            CType H5T_C_S1;
        }
        DATASPACE SCALAR
        DATA {
            (0): "degrees"
        }
    }
    # ... other attributes
}
```

h5dump of a NeXus attribute

```hdf5
ATTRIBUTE "axes" {
    DATATYPE H5T_STRING {
        STRSIZE 9;
        STRPAD H5T_STR_NULLPAD;
        CSET H5T_CSET_ASCII;
        CType H5T_C_S1;
    }
    DATASPACE SCALAR
    DATA {
        (0): "two_theta"
    }
}
```
# NeXus links have two parts in HDF5 files.
# The dataset is created in some group.
# A "target" attribute is added to indicate the HDF5 path to this dataset.

ATTRIBUTE "target" {
    DATATYPE H5T_STRING {
        STRSIZE 21;
        STRPAD H5T_STR_NULLPAD;
        CSET H5T_CSET_ASCII;
        CTYPE H5T_C_S1;
    }
    DATASPACE SCALAR
    DATA {
        (0): "/entry/data/two_theta"
    }
}

# then, the hard link is created that refers to the original dataset
# (Since the name is "two_theta" in this example, it is understood that
# this link is created in a different HDF5 group than "/entry/data".)

DATASET "two_theta" {
    HARDLINK "/entry/data/two_theta"
}

## 1.3 Constructing NeXus Files and Application Definitions

In NeXus Design, we discussed the design of the NeXus format in general terms. In this section a more tutorial style introduction in how to construct a NeXus file is given. As an example a hypothetical instrument named WONI will be used.

**Note:** If you are looking for a tutorial on reading or writing NeXus data files using the NeXus API, consult the NAPI: NeXus Application Programmer Interface (frozen) chapter. For code examples (with or without NAPI), refer to the Code Examples in Various Languages chapter.

### 1.3.1 The WOnderful New Instrument (WONI)

Consider yourself to be responsible for some hypothetical WOnderful New Instrument (WONI). You are tasked to ensure that WONI will record data according to the NeXus standard. For the sake of simplicity, WONI bears a strong resemblance to a simple powder diffractometer, but let's pretend that WONI cannot use any of the existing NXDL application definitions.

WONI uses collimators and a monochromator to illuminate the sample with neutrons of a selected wavelength as described in The (fictional) WONI example powder diffractometer. The diffracted beam is collected in a large, banana-shaped, position sensitive detector. Typical data looks like Example Powder Diffraction Plot from (fictional) WONI at HYNES. There is a generous background to the data plus quite a number of diffraction peaks.
Fig. 15: The (fictional) WONI example powder diffractometer
Fig. 16: Example Powder Diffraction Plot from (fictional) WONI at HYNES
1.3.2 Constructing a NeXus file for WONI

The starting point for a NeXus file for WONI will be an empty basic NeXus file hierarchy as documented in the next figure. In order to arrive at a full NeXus file, the following steps are required:

1. For each instrument component, decide which parameters need to be stored
2. Map the component parameters to NeXus groups and parameters and add the component to the NXinstrument hierarchy
3. Decide what needs to go into NXdata. While this group is optional, you are urged strongly to provide an NXdata group to support default plotting.
4. Fill the NXsample and NXmonitor groups

Basic structure of a NeXus file

```
entry: NXentry
   NXdata
   NXinstrument
   NXmonitor
   NXsample
```

Decide which parameters need to be stored

Now the various groups of this empty NeXus file shell need to be filled. The next step is to look at a design drawing of WONI. Identify all the instrument components like collimators, detectors, monochromators etc. For each component decide which values need to be stored. As NeXus aims to describe the experiment as good as possible, strive to capture as much information as practical.

Mapping parameters to NeXus

With the list of parameters to store for each component, consult the reference manual section on the NeXus base classes. You will find that for each of your instruments components there will be a suitable NeXus base class. Add this base class together with a name as a group under NXinstrument in your NeXus file hierarchy. Then consult the possible parameter names in the NeXus base class and match them with the parameters you wish to store for your instruments components.

As an example, consider the monochromator. You may wish to store: the wavelength, the d-value of the reflection used, the type of the monochromator and its angle towards the incoming beam. The reference manual tells you that NXcrystal is the right base class to use. Suitable fields for your parameters can be found in there to. After adding them to the basic NeXus file, the file looks like in the next figure:

Basic structure of a NeXus file with a monochromator added

```
entry: NXentry
   NXdata
   NXinstrument
       monochromator: NXcrystal
           wavelength
           d_spacing
           rotation_angle
```
If a parameter or even a whole group is missing in order to describe your experiment, do not despair! Contact the NIAC and suggest to add the group or parameter. Give a little documentation what it is for. The NIAC will check that your suggestion is no duplicate and sufficiently documented and will then proceed to enhance the base classes with your suggestion.

A more elaborate example of the mapping process is given in the section *Creating a NXDL Specification*.

**Decide on NXdata**

The `NXdata/` group is supposed to contain the data required to put up a quick plot. For WONI this is a plot of counts versus two theta (polar_angle in NeXus) as can be seen in *Example Powder Diffraction Plot from (fictional) WONI at HYNES*. Now, in `NXdata`, create links to the appropriate data items in the `NXinstrument` hierarchy. In the case of WONI, both parameters live in the `detector:NXdetector` group.

**Fill in auxiliary Information**

Look at the section on `NXsample` in the NeXus reference manual. Choose appropriate parameters to store for your samples. Probably at least the name will be needed.

In order to normalize various experimental runs against each other it is necessary to know about the counting conditions and especially the monitor counts of the monitor used for normalization. The NeXus convention is to store such information in a `control:NXmonitor` group at `NXentry` level. Consult the reference for `NXmonitor` for field names. If additional monitors exist within your experiment, they will be stored as additional `NXmonitor` groups at entry level.

Consult the documentation for `NXentry` in order to find out under which names to store information such as titles, user names, experiment times etc.

A more elaborate example of this process can be found in the following section on creating an application definition.

### 1.3.3 Creating a NXDL Specification

An NXDL specification for a NeXus file is required if you desire to standardize NeXus files from various sources. Another name for a NXDL description is application definition. A NXDL specification can be used to verify NeXus files to conform to the standard encapsulated in the application definition. The process for constructing a NXDL specification is similar to the one described above for the construction of NeXus files.

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 …
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

**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 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 *Example Powder Diffraction Plot from (fictional) WONI at HYNES* above. Summarizing, the basic information required from WONI is given next.

- **title** of measurement
- **sample name**
- **sample temperature**
- counts from the incident beam **monitor**
- type of radiation **probe**
- **wavelength** ($\lambda$) of radiation incident on sample
- angle ($2\theta$ or **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 (*Using an Application Definition*) will reveal the secret how to go from an application definition to a practical file.
Step 2: Map Data into the NeXus Hierarchy

This step is actually easier than 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 (The (fictional) WONI example powder diffractometer), 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 the next table:
Looking at this table, 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 most things which possibly can occur. You do not have to give all that information. Keep it simple and only require data that is needed for typical data analysis for this type of application.

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 altogether. New fields and base classes can be introduced if necessary.

In any case please feel free to contact the NIAC via the mailing list with questions or suggestions.

**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 the XML code below.

**Note:** This is just the basic XML for a NXDL definition. It is advisable to change some of the documentation strings.

### NXDL template file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!--
# NeXus - Neutron and X-ray Common Data Format
#
# Copyright (C) 2008-2022 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 3 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

(continues on next page)```
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.

The next thing which needs to be done is adding groups into the definition. A group is defined by some XML, as in this example:

```
<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 similar to this:

```
<field name="polar_angle" type="NX_FLOAT units="NX_ANGLE">
  <doc>Link to polar angle in /NXentry/NXinstrument/NXdetector</doc>
  <dimensions rank="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: rank="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 Full listing of the WONI Application Definition. 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.
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.

Full listing of the WONI Application Definition

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="nxdlformat.xsl" ?>
<!--
 # NeXus - Neutron and X-ray Common Data Format
 #
 # Copyright (C) 2008-2022 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 3 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
 -->
<definition name="NXmonopd" 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">
    <symbols>
        <doc>
The symbol(s) listed here will be used below to coordinate datasets with the same shape.
</doc>
        <symbol name="i">
            <doc>i is the number of wavelengths</doc>
        </symbol>
    </symbols>
</definition>
```

(continues on next page)
In this 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.

```xml
<group type="NXentry" name="entry">
  <field name="title"/>
  <field name="start_time" type="NX_DATE_TIME"/>
  <field name="definition">
    <doc>Official NeXus NXDL schema to which this file conforms</doc>
    <enumeration>
      <item value="NXmonopd"/>
    </enumeration>
  </field>
</group>
```

```xml
<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 rank="1">
        <dim index="1" value="i"/>
      </dimensions>
    </field>
  </group>
  <group type="NXdetector">
    <field name="polar_angle" type="NX_FLOAT" axis="1">
      <dimensions rank="1">
        <dim index="1" value="nDet"/>
      </dimensions>
    </field>
    <field name="data" type="NX_INT" signal="1">
      <doc>detector signal (usually counts) are already</doc>
    </field>
  </group>
</group>
```
corrected for detector efficiency
</doc>
<dimensions rank="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>
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 (see the regular expression pattern for NXDL group and field names in the Naming Conventions section). 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 to save space. Application definitions describe only what is included within an NXentry and so have no power to enforce any particular usage of NXentry groups. However, documentation within and accompanying an application definition can provide guidance and recommendations on situations where the use of multiple NXentry groups would be appropriate.

1.3.4 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.

1.4 Strategies for storing information in NeXus data files

NeXus may appear daunting, at first, to use. The number of base classes is quite large as well as is the number of application definitions. This chapter describes some of the strategies that have been recommended for how to store information in NeXus data files.

When we use the term storing, some might be helped if they consider this as descriptions for how to classify their data.

It is intended for this chapter to grow, with the addition of different use cases as they are presented for suggestions.
1.4.1 Strategies: The simplest case(s)

Perhaps the simplest case might be either a step scan with two or more columns of data. Another simple case might be a single image acquired by an area detector. In either of these hypothetical cases, the situation is so simple that there is little additional information available to be described (for whatever reason).

**Step scan with two or more data columns**

Consider the case where we wish to store the data from a step scan. This case may involve two or more related 1-D arrays of data to be saved, each having the same length. For our hypothetical case, we'll have these positioners as arrays and assume that a default plot of photodiode vs. ar:

<table>
<thead>
<tr>
<th>positioner arrays</th>
<th>detector arrays</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar, ay, dy</td>
<td>I0, I00, time, Epoch, photodiode</td>
</tr>
</tbody>
</table>

**Data file structure for Step scan with two or more data columns**

```plaintext
file.nxs: NeXus HDF5 data file
 @default = "entry"
entry: NXentry
 @NX_class = "NXentry"
 @default = "data"
data: NXdata
 @NX_class = "NXdata"
 @signal = "photodiode"
 @axes = "ar"
ar: NX_FLOAT[]
ay: NX_FLOAT[]
dy: NX_FLOAT[]
I0: NX_FLOAT[]
I00: NX_FLOAT[]
time: NX_FLOAT[]
Epoch: NX_FLOAT[]
photodiode: NX_FLOAT[]
```

1.4.2 Strategies: The wavelength

*Where should the wavelength of my experiment be written?* This is one of the *Frequently Asked Questions*. The canonical location to store wavelength has been:

/NXentry/NXinstrument/NXcrystal/wavelength
Partial data file structure for *canonical location to store wavelength*

```
entry: NXentry
    @NX_class = NXentry
    instrument: NXinstrument
        @NX_class = NXinstrument
        crystal: NXcrystal
            @NX_class = NXcrystal
            wavelength: NX_FLOAT
```

More recently, this location makes more sense to many:

```
/NXentry/NXinstrument/NXmonochromator/wavelength
```

Partial data file structure for *location which makes more sense to many to store wavelength*

```
entry: NXentry
    @NX_class = NXentry
    instrument: NXinstrument
        @NX_class = NXinstrument
        monochromator: NXmonochromator
            @NX_class = NXmonochromator
            wavelength: NX_FLOAT
```

*NXcrystal* describes a crystal monochromator or analyzer. Recently, scientists with monochromatic radiation not defined by a crystal, such as from an electron-beam undulator or a neutron helical velocity selector, were not satisfied with creating a fictitious instance of a crystal just to preserve the wavelength from their instrument. Thus, the addition of the *NXmonochromator* base class to NeXus, which also allows “energy” to be specified if one is so inclined.

**Note:** See the *Class path specification* section for a short discussion of the difference between the HDF5 path and the NeXus symbolic class path.

### 1.4.3 Strategies: Time-stamped data

*How should I store time-stamped data?*

Time-stamped data can be stored in either *NXlog* and *NXevent_data* structures. Of the two, *NXlog* is the most important one, *NXevent_data* is normally only used for storing detector time of flight event data and *NXlog* would be used for storing any other time-stamped data, e.g. sample temperature, chopper top-dead-centre, motor position, detector images etc.

Regarding the NeXus file structure to use, there is one simple rule: just use the standard NeXus file structure but insert/replace the fields for streamed data elements through *NXlog* or *NXevent_data* structures. For example, consider the collection of detector images against a change in the magnetic field on the sample. Then, both NXsample/magnetic_field and NXdetector/data would be *NXlog* structures containing the time stamped data.

Both *NXlog* and *NXevent_data* have additional support for storing time-stamped data in the form of cues; cues can be used to place markers in the data that allow one to quickly look up coarse time ranges of interest. This coarse range of data can then be manually trimmed to be more selective, if required. The application writing the NeXus file is responsible for writing cues and when they are written. For example, the cue could be written every 10 seconds, every pulse, every 100 datapoints and so on.
Let’s consider the case where NXlog is being used to store sample temperature data that has been sampled once every three seconds. The application that wrote the data has added cues every 20 seconds. Pictorially, this may look something like this:

If we wanted to retrieve the mean temperature between 30 and 40 seconds, we would use the cues to grab the data between 20 seconds and 40 seconds, and then trim that data to get the data we want. Obviously in this simple example this does not gain us a lot, but it is easy to see that in a large dataset having appropriately placed cues can save significant computational time when looking up values in a certain time-stamp range. NeXus has actually borrowed the cueing table concept from video file formats where it allows viewing software to quickly access your favourite scene. Correspondingly, cueing in NeXus allows you to quickly access your favourite morsel of time stamped data.

In the NeXus Features repository, the feature ECB064453EDB096D shows example code that uses cues to select time-stamped data.
1.4.4 Strategies: The next case

The NIAC: The NeXus International Advisory Committee welcomes suggestions for additional sections in this chapter.

1.5 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. 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.

1.5.1 nxvalidate

NeXus validation tool written in C (not via NAPI).

Its dependencies are libxml2 and the HDF5 libraries, version 1.8.9 or better. Its purpose is to validate HDF5 files against NeXus application definitions.

See the program documentation for more details: https://github.com/nexusformat/cnxvalidate.git

1.5.2 punx

Python Utilities for NeXus HDF5 files

punx can validate both NXDL files and NeXus HDF5 data files, as well as print the structure of any HDF5 file, even non-NeXus files.

NOTE: project is under initial construction, not yet released for public use, but is useful in its present form (version 0.2.5).

punx can show the tree structure of any HDF5 file. The output is more concise than that from h5dump.

See the program documentation for more details: https://punx.readthedocs.io

1.6 Frequently Asked Questions

This is a list of commonly asked questions concerning the NeXus data format.

1. Is it Nexus, NeXus or NeXuS?

NeXus is correct. It is a format for data from Neutron and X-ray facilities, hence those first letters are capitalised. The format is also used for muon experiments, but there is no mu (or m) in NeXus and no s in muon. So the s stays in lower case.

2. How many facilities use NeXus?
This is not easy to say, not all facilities using NeXus actively participate in the committee. Some facilities have reported their adoption status on the Facilities web page. Please have a look at this list. Keep in mind that it is never fully complete or up to date.

3. NeXus files are binary? This is crazy! How am I supposed to see my data?

Various tools are listed in the NeXus Utilities section to inspect NeXus data files. The easiest graphical tool to use is HDFview which can open any HDF file. Other tools such as PyMCA and NeXPy provide visualization of scientific data while h5dump and punx tree provide text renditions of content and structure. If you want to try, for example nxbrowse is a utility provided by the NeXus community that can be very helpful to those who want to inspect their files and avoid graphical applications. For larger data volumes the binary backends used with the appropriate tools are by far superior in terms of efficiency and speed and most users happily accept that after having worked with supersized “human readable” files for a while.

4. What on-disk file format should I choose for my data?

HDF5 is the default file container to use for NeXus data. It is the recommended format for all applications. HDF4 is still supported as a on disk format for NeXus but for new installations preference should be given to HDF5.

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. It is not compulsory to include every item that has been defined in the base class if it is not relevant to your experiment. On the other hand, a NeXus application definition lists a smaller set of compulsory items that should allow other researchers or software to analyze your data. You should really follow the application definition that corresponds to your experiment to take full advantage of NeXus.

6. I don’t like NeXus. It seems much faster and simpler to develop my own file format. Why should I even consider NeXus?

If you consider using an efficient on disk storage format, HDF5 is a better choice than most others. It is fast and efficient and well supported in all mainstream programming languages and a fair share of popular analysis packages. The format is so widely used and backed by a big organisation that it will continue to be supported for the foreseeable future. So if you are going to use HDF5 anyway, why not use the NeXus definition to lay out the data in a standardised way? The NeXus community spent years trying to get the standard right and while you will not agree with every single choice they made in the past, you should be able to store the data you have in a quite reasonable way. If you do not comply with NeXus, chances are most people will perceive your format as different but not necessarily better than NeXus by any large measure. So it may not be worth the effort. Seriously.

If you encounter any problems because the classes are not sufficient to describe your experiment, please contact the mailing list. Pull requests for the definitions repository (for example adding contributed definitions) are also welcome (see next question). The NIAC is always willing to consider new proposals.

7. I want to contribute an application definition.

How do I go about it?

Read the NXDL Tutorial in Creating a NXDL Specification and have a try. You can ask for help on the mailing lists. Once you have a definition that is working well for at least your case, you can submit it to the NIAC for acceptance as a standard. The procedures for acceptance are defined in the NIAC constitution.

8. What is the purpose of NXdata?

Refer to the most recent version of the NIAC constitution on the NIAC web page: https://www.nexusformat.org/NIAC.html#constitution

1.6. Frequently Asked Questions
NXdata identifies the default plottable data. This is one of the basic motivations (see Simple plotting) for the NeXus standard. The choice of the name NXdata is historic and does not really reflect its function. The NXdata group contains data or links to the data stored elsewhere.

9. How do I identify the plottable data?

See the section: Find the plottable data.

10. 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.

11. 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.

12. Do I have to use the NAPI subroutines? Can’t I read (or write) the NeXus data files with my own routines?

You are not required to use the NAPI to write valid NeXus data files. It is possible to avoid the NAPI to write and read valid NeXus data files. But, the programmer who chooses this path must have more understanding of how the NeXus HDF data file is written. Validation of data files written without the NAPI is strongly encouraged.

13. I’m using links to place data in two places. Which one should be the data and which one is the link?

Note: NeXus uses HDF5 hard links

In HDF, a hard link points to a data object. A soft link points to a directory entry. Since NeXus uses hard links, there is no need to distinguish between two (or more) directory entries that point to the same data.

Both places have pointers to the actual data. That is the way hard links work in HDF5. There is no need for a preference to either location. NeXus defines a target attribute to label one directory entry as the source of the data (in this, the link target). This has value in only a few situations such as when converting the data from one format to another. By identifying the original in place, duplicate copies of the data are not converted.

14. If I write my data according to the current specification for NXsas
(substitute any other application definition), will other software be able to read my data?

Yes. NXsas, like other Application Definitions, defines and names the minimum information required for analysis or data processing. As long as all the information required by the specification is present, analysis software should be able to process the data. If other information is also present, there is no guarantee that small-angle scattering analysis software will notice.

15. Where do I store the wavelength of my experiment?

See the Strategies: The wavelength section.

16. Where do I store metadata about my experiment?

See the Where to Store Metadata section.

17. What file extension should I use when writing a NeXus data file?
Any extension is permitted. Common extensions are .h5, .hdf, .hdf5, and .nxs while others are possible. See the many examples in the NeXus exampledata repository. (https://github.com/nexusformat/exampledata)

18. Can instances of classes inside definitions require new fields that were previously optional?
   Yes. That is one of the motivations to have application definitions. By default, all content in an application definition is required.
   For example, the radiation field in NXcanSAS requires 1 (and only 1) instance.

19. Can instances of classes inside definitions make optional new fields that were previously not mentioned?
   Yes. To make it optional, set attribute minOccurs="0".
   For example, see the Idev field in NXcanSAS.

20. Can instances of classes inside definitions require new fields that were previously not mentioned?
   Yes.
   For example, see the qx field in NXiqproc.

21. Can we view the process of defining classes within an application definition as defining a subclass of the original class? That is, all instances of the class within the definition are valid instances of the original class, but not vice-versa?
   Keep in mind that NeXus is not specifically object oriented. The putative super class might be either NXentry (for single-technique data, such as SAXS) or NXsubentry (for multi-technique data such as SAXS/WAXS/USAXS/GIWAXS or SAXS/SANS).
   If you are thinking of a new application definition that uses another as a starting point (like a super class), then there is an extends attribute in the definition element of the NXDL file (example here from NXarpes):

```xml
<definition name="NXarpes" extends="NXobject" type="group"
```

which describes this relationship. For most (?all?) all NXDL files to date, they extend the NXobject base class (the base object of NeXus).
EXAMPLES OF WRITING AND READING NEXUS DATA FILES

Simple examples of reading and writing NeXus data files are provided in the NeXus Introduction chapter and also in the NAPI: NeXus Application Programmer Interface (frozen) chapter.

2.1 Code Examples in Various Languages

Each example in this section demonstrates writing and reading NeXus compliant files in various languages with different libraries. Most examples are using the HDF5 file format. Note however that other container formats like the legacy format HDF4 or XML can also be used to store NeXus compliant data.

Please be aware that not all examples are up to date with the latest format recommendations.

2.1.1 HDF5 in C with libhdf5

C-language code examples are provided for writing and reading NeXus-compliant files using the native HDF5 interfaces. These examples are derived from the simple NAPI examples for writing and reading given in the Introduction chapter.

Writing a simple NeXus file using native HDF5 commands in C

**Note:** This example uses the new method described in Associating plottable data using attributes applied to the NXdata group for indicating plottable data.

```c
/**
 * This is an example how to write a valid NeXus file
 * using the HDF-5 API alone. This structure which is
 * going to be created is:
 *
 * scan:NXentry
 * @signal = "counts"
 * @axes = "two_theta"
 * @two_theta_indices = 0
 * counts[]
 * @units="counts"
 * two_theta[]
 * @units="degrees"
 */
```

(continues on next page)
* WARNING: each of the HDF function below needs to be
* wrapped into something like:
* if((hdfid = H5function(...)) < 0){
* handle error gracefully
* }
* I left the error checking out in order to keep the
* code clearer
* This also installs a link from /scan/data/two_theta to /scan/hugo
* Mark Koennecke, October 2011
*/
#include <hdf5.h>
#include <stdlib.h>
#include <string.h>

static void write_string_attr(hid_t hid, const char* name, const char* value)
{
  /* HDF-5 handles */
  hid_t atts, atttype, attid;
  atts = H5Screate(H5S_SCALAR);
  atttype = H5Tcopy(H5T_C_S1);
  H5Tset_size(atttype, strlen(value));
  attid = H5Acreate(hid, name, atttype, atts, H5P_DEFAULT, H5P_DEFAULT);
  H5Awrite(attid, atttype, value);
  H5Sclose(atts);
  H5Tclose(atttype);
  H5Aclose(attid);
}

static void write_int_attr(hid_t hid, const char* name, int value)
{
  /* HDF-5 handles */
  hid_t atts, atttype, attid;
  atts = H5Screate(H5S_SCALAR);
  atttype = H5Tcopy(H5T_NATIVE_INT);
  H5Tset_size(atttype, 1);
  attid = H5Acreate(hid, name, atttype, atts, H5P_DEFAULT, H5P_DEFAULT);
  H5Awrite(attid, atttype, &value);
  H5Sclose(atts);
  H5Tclose(atttype);
  H5Aclose(attid);
}

#define LENGTH 400
int main(int argc, char *argv[])
{
  float two_theta[LENGTH];
}
```c
int counts[LENGTH], i, rank;

/* HDF-5 handles */
hid_t fid, fapl, gid;
hid_t datatype, dataspace, dataprop, dataid;
hsize_t dim[1], maxdim[1];

/* create some data: nothing NeXus or HDF-5 specific */
for(i = 0; i < LENGTH; i++){
    two_theta[i] = 10. + .1*i;
    counts[i] = (int)(1000 * ((float)random()/(float)RAND_MAX));
}
dim[0] = LENGTH;
maxdim[0] = LENGTH;
rank = 1;

/* open the file. The file attribute forces normal file
* closing behaviour down HDF-5's throat */
fapl = H5Pcreate(H5P_FILE_ACCESS);
H5Pset_fclose_degree(fapl,H5F_CLOSE_STRONG);
 fid = H5Fcreate("NXfile.h5", H5F_ACC_TRUNC, H5P_DEFAULT,fapl);
 H5Pclose(fapl);

/* create scan:NXentry */
gid = H5Gcreate(fid, "scan",H5P_DEFAULT,H5P_DEFAULT,H5P_DEFAULT);
/* store the NX_class attribute. Notice that you
* have to take care to close those hids after use */
write_string_attr(gid, "NX_class", "NXentry");

/* same thing for data:Nxdata in scan:NXentry. */
gid = H5Gcreate(fid, "/scan/data",H5P_DEFAULT,H5P_DEFAULT,H5P_DEFAULT);
write_string_attr(gid, "NX_class", "NXdata");

/* define axes. */
write_string_attr(gid, "signal", "counts");
write_string_attr(gid, "axes", "two_theta");
write_int_attr(gid, "two_theta_indices", 0);
```

(continues on next page)
/*
 * store the counts dataset
 */
dataspace = H5Screate_simple(rank,dim,maxdim);
datatype = H5Tcopy(H5T_NATIVE_INT);
dataprop = H5Pcreate(H5P_DATASET_CREATE);
dataid = H5Dcreate(gid,"counts",datatype,dataspace,H5P_DEFAULT,dataprop,H5P_DEFAULT);
H5Dwrite(dataid, datatype, H5S_ALL, H5S_ALL, H5P_DEFAULT, counts);
H5Sclose(dataspace);
H5Tclose(datatype);
H5Pclose(dataprop);

/*
 * set the units attribute
 */
write_string_attr(dataid, "units", "counts");
H5Dclose(dataid);

/*
 * store the two_theta dataset
 */
dataspace = H5Screate_simple(rank,dim,maxdim);
datatype = H5Tcopy(H5T_NATIVE_FLOAT);
dataprop = H5Pcreate(H5P_DATASET_CREATE);
dataid = H5Dcreate(gid,"two_theta",datatype,dataspace,H5P_DEFAULT,dataprop,H5P_DEFAULT);
H5Dwrite(dataid, datatype, H5S_ALL, H5S_ALL, H5P_DEFAULT, two_theta);
H5Sclose(dataspace);
H5Tclose(datatype);
H5Pclose(dataprop);

/*
 * set the units attribute
 */
write_string_attr(dataid, "units", "degrees");

/*
 * set the target attribute for linking
 */
write_string_attr(dataid, "target", "/scan/data/two_theta");
H5Dclose(dataid);

/*
 * make a link in /scan to /scan/data/two_theta, thereby
 * renaming two_theta to hugo
 */
H5Glink(fid,H5G_LINK_HARD,"/scan/data/two_theta","/scan/hugo");

/*
 * close the file
 */
(continues on next page)
Reading a simple NeXus file using native HDF5 commands in C

```c
/**
 * Reading example for reading NeXus files with plain
 * HDF-5 API calls. This reads out counts and two_theta
 * out of the file generated by nxh5write.
 * WARNING: I left out all error checking in this example.
 * In production code you have to take care of those errors
 * Mark Koennecke, October 2011
 */
#include <hdf5.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    float *two_theta = NULL;
    int *counts = NULL, rank, i;
    hid_t fid, dataid, fapl;
    hsize_t *dim = NULL;
    hid_t dataspace, memdataspace;
    
    /*
     * Open file, thereby enforcing proper file close
     * semantics
     */
    fapl = H5Pcreate(H5P_FILE_ACCESS);
    H5Pset_fclose_degree(fapl, H5F_CLOSE_STRONG);
    fid = H5Fopen("NXfile.h5", H5F_ACC_RDONLY, fapl);
    H5Pclose(fapl);
    
    /*
     * open and read the counts dataset
     */
    dataid = H5Dopen(fid, "/scan/data/counts", H5P_DEFAULT);
    dataspace = H5Sget_space(dataid);
    rank = H5Sget_simple_extent_ndims(dataspace);
    dim = malloc(rank*sizeof(hsize_t));
    H5Sget_simple_extent_dims(dataspace, dim, NULL);
    counts = malloc(dim[0]*sizeof(int));
    memdataspace = H5Tcopy(H5T_NATIVE_INT32);
    H5Dread(dataid, memdataspace, H5S_ALL, H5S_ALL, H5P_DEFAULT, counts);
    H5Dclose(dataid);
    H5Sclose(dataspace);
    H5Tclose(memdataspace);
    return 0;
}
```

(continues on next page)
OPEN AND READ THE TWO_Theta DATA SET

46 /*
47 * open and read the two_theta data set
48 */
49 
dataid = H5Dopen(fid,"/scan/data/two_theta",H5P_DEFAULT);
50 dataspace = H5Dget_space(dataid);
51 rank = H5Sget_simple_extent_ndims(dataspace);
52 dim = malloc(rank*sizeof(hsize_t));
53 H5Sget_simple_extent_dims(dataspace, dim, NULL);
54 two_theta = malloc(dim[0]*sizeof(float));
55 memdataspace = H5Tcopy(H5T_NATIVE_FLOAT);
56 H5Dread(dataid,memdataspace,H5S_ALL, H5S_ALL,H5P_DEFAULT, two_theta);
57 H5Dclose(dataid);
58 H5Sclose(dataspace);
59 H5Tclose(memdataspace);
60 
61 for(i = 0; i < dim[0]; i++){
62     printf("%8.2f %10d\n", two_theta[i], counts[i]);
63 }
64 
65 H5Fclose(fid);

2.1.2 HDF5 in Python

One way to gain a quick familiarity with NeXus is to start working with some data. For at least the first few examples in this section, we have a simple two-column set of 1-D data, collected as part of a series of alignment scans by the Advanced Photon Source USAXS instrument during the time it was stationed at beam line 32ID. We will show how to read and write this data in Python using both the `nexusformat` and `h5py` packages. The `nexusformat` package provides a simplified syntax for reading and writing NeXus-compliant files by automatically handling some of the features required by the NeXus standard, such as the attributes that define group classes and plottable data. However, it also uses the `h5py` package to read/write the HDF5 files on disk. We provide tabbed examples showing how to produce equivalent files either using `nexusformat` or directly in `h5py`.

The actual data to be written was extracted (elsewhere) from a spec\(^1\) data file and read as a text block from a file by the Python source code. Our examples will start with the simplest case and add only mild complexity with each new case since these examples are meant for those who are unfamiliar with NeXus.

---

1 `nexusformat`: https://nexpy.github.io/nexpy/
2 `h5py`: https://www.h5py.org/
3 SPEC: http://certif.com/spec.html
Getting started

Write a NeXus HDF5 File

In the main code section of `simple_example_basic_write.py`, the data (mr is similar to “two_theta” and I00 is similar to “counts”) is collated into two Python lists. We use the `numpy` package to read the file and parse the two-column format.

The new HDF5 file is opened (and created if not already existing) for writing, setting common NeXus attributes in the same command from our support library. Proper HDF5+NeXus groups are created for `/entry:NXentry/mr_scan:NXdata`. Since we are not using the NAPI, our support library must create and set the `NX_class` attribute on each group.

**Note:** We want to create the desired structure of `/entry:NXentry/mr_scan:NXdata/`.

1. First, our support library calls `f = h5py.File()` to create the file and root level NeXus structure.
2. Then, it calls `nxentry = f.create_group("entry")` to create the `NXentry` group called `entry` at the root level.
3. Then, it calls `nxdata = nxentry.create_group("mr_scan")` to create the `NXentry` group called `entry` as a child of the `NXentry` group.

Next, we create a dataset called `title` to hold a title string that can appear on the default plot.

Next, we create datasets for `mr` and `I00` using our support library. The data type of each, as represented in `numpy`, will be recognized by `h5py` and automatically converted to the proper HDF5 type in the file. A Python dictionary of attributes is given, specifying the engineering units and other values needed by NeXus to provide a default plot of this data. By setting `signal="I00"` as an attribute on the group, NeXus recognizes `I00` as the default y axis for the plot. The `axes="mr"` attribute on the `NXdata` group connects the dataset to be used as the x axis.

Finally, we **must** remember to call `f.close()` or we might corrupt the file when the program quits.

**simple_example_basic_write.py:** Write a NeXus HDF5 file using Python with h5py

```python
#!/usr/bin/env python
"""Wright a NeXus HDF5 file using h5py and numpy""

from pathlib import Path
from re import X
import numpy

from nexusformat.nexus import NXdata, NXentry, NXfield, nxopen

print("Write a NeXus HDF5 file")
fileName = "simple_example_basic.nexus.hdf5"

# load data from two column format
data_filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
data = numpy.loadtxt(data_filename).T
```
mr_arr = data[0]
i00_arr = numpy.asarray(data[1], "int32")

# create the HDF5 NeXus file
with nxopen(fileName, "w") as f:
    # create the NXentry group
    f["entry"] = NXentry()
    f["entry/title"] = "1-D scan of I00 v. mr"
    # create the NXdata group
    x = NXfield(mr_arr, name="mr", units="degrees", long_name="USAXS mr (degrees)")
y = NXfield(i00_arr, name="I00", units="counts",
               long_name="USAXS I00 (counts)")
f["entry/mr_scan"] = NXdata(y, x)

print("wrote file:", fileName)
nxentry = f.create_group("entry")
xnentry.attrs["NX_class"] = "NXentry"
xnentry.attrs["default"] = "mr_scan"
xnentry.create_dataset("title", data="1-D scan of I00 v. mr")

# create the NXentry group
nxdata = xnentry.create_group("mr_scan")

nxdata.attrs["NX_class"] = "NXdata"
xnentry.attrs["signal"] = "I00"  # Y axis of default plot
nxdata.attrs["axes"] = "mr"  # X axis of default plot
nxdata.attrs["mr_indices"] = [0, ]  # use "mr" as the first dimension of I00

# X axis data
ds = nxdata.create_dataset("mr", data=mr_arr)
ds.attrs["units"] = "degrees"
ds.attrs["long_name"] = "USAXS mr (degrees)"  # suggested X axis plot label

# Y axis data
ds = nxdata.create_dataset("I00", data=i00_arr)
ds.attrs["units"] = "counts"
ds.attrs["long_name"] = "USAXS I00 (counts)"  # suggested Y axis plot label

print("wrote file:", fileName)

Read a NeXus HDF5 File

The file reader, simple_example_basic_read.py, opens the HDF5 we wrote above, prints the HDF5 attributes from the file, reads the two datasets, and then prints them out as columns. As simple as that. Of course, real code might add some error-handling and extracting other useful stuff from the file.

Note: See that we identified each of the two datasets using HDF5 absolute path references (just using the group and dataset names). Also, while coding this example, we were reminded that HDF5 is sensitive to upper or lowercase. That is, I00 is not the same is i00.

simple_example_basic_read.py: Read a NeXus HDF5 file using Python

nexusformat

#!/usr/bin/env python
"""Reads NeXus HDF5 files using nexusformat and prints the contents"""

from nexusformat.nexus import nxopen

fileName = "simple_example_basic.nexus.hdf5"
with nxopen(fileName) as f:
    print(f.tree)
#!/usr/bin/env python
"""Reads NeXus HDF5 files using h5py and prints the contents"""

import h5py  # HDF5 support

fileName = "simple_example_basic.nexus.hdf5"
with h5py.File(fileName, "r") as f:
    for item in f.attrs.keys():
        print(item + ":", f.attrs[item])
    mr = f["/entry/mr_scan/mr"]
    i00 = f["/entry/mr_scan/I00"]
    print("%s\t%s\t%s" % ("#", "mr", "I00"))
    for i in range(len(mr)):
        print("%d\t%g\t%d" % (i, mr[i], i00[i]))

Output from simple_example_basic_read.py is shown next.

Output from simple_example_basic_read.py

file_name: simple_example_basic.nexus.hdf5
file_time: 2010-10-18T17:17:04-0500
creator: simple_example_basic_write.py
HDF5_Version: 1.8.5
NeXus_version: 4.3.0
h5py_version: 1.2.1
instrument: APS USAXS at 32ID-B
#  mr  I00
0  17.9261 1037
1  17.9259 1318
2  17.9258 1704
3  17.9256 2857
4  17.9254 4516
5  17.9252 9998
6  17.9251 23819
7  17.9249 31662
8  17.9247 40458
9  17.9246 49087
10 17.9244 56514
11 17.9243 63499
12 17.9241 66802
13 17.9239 66863
14 17.9237 66599
15 17.9236 66206
16 17.9234 65747
17 17.9232 65250
18 17.9231 64129
19 17.9229 63044
20 17.9228 60796
21 17.9226 56795
22 17.9224 51550
(continues on next page)
The Python code and files related to this section may be downloaded from the following table.

<table>
<thead>
<tr>
<th>file</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>../simple_example.dat</td>
<td>2-column ASCII data used in this section</td>
</tr>
<tr>
<td>simple_example_basic_read.py</td>
<td>h5py code to read example simple_example_basic.nexus.hdf5</td>
</tr>
<tr>
<td>nexusformat/simple_example_basic_read.py</td>
<td>nexusformat code to read example simple_example_basic.nexus.hdf5</td>
</tr>
<tr>
<td>simple_example_basic_write.py</td>
<td>h5py code to write example simple_example_basic.nexus.hdf5</td>
</tr>
<tr>
<td>nexusformat/simple_example_basic_write.py</td>
<td>nexusformat code to write example simple_example_basic.nexus.hdf5</td>
</tr>
<tr>
<td>simple_example_basic.nexus_h5dump.txt</td>
<td>h5dump analysis of the NeXus file</td>
</tr>
<tr>
<td>simple_example_basic.nexus.hdf5</td>
<td>NeXus file written by BasicWriter</td>
</tr>
<tr>
<td>simple_example_basic.nexus_structure.txt</td>
<td>punx tree analysis of the NeXus file</td>
</tr>
</tbody>
</table>

### Write a NeXus HDF5 file

In this example, the 1-D scan data will be written into the simplest possible NeXus HDF5 data file, containing only the required NeXus components. NeXus requires at least one NXentry group at the root level of an HDF5 file. The NXentry group contains all the data and associated information that comprise a single measurement. NXdata is used to describe the plottable data in the NXentry group. The simplest place to store data in a NeXus file is directly in the NXdata group, as shown in the next figure.

In the above figure, the data file (simple_example_write1_h5py.hdf5) contains a hierarchy of items, starting with an NXentry named entry. (The full HDF5 path reference, /entry in this case, is shown to the right of each component in the data structure.) The next h5py code example will show how to build an HDF5 data file with this structure. Starting with the numerical data described above, the only information written to the file is the absolute minimum information NeXus requires. In this example, you can see how the HDF5 file is created, how Groups and datasets (Fields) are created, and how Attributes are assigned. Note particularly the NX_class attribute on each HDF5 group that describes which of the NeXus Base Class Definitions is being used. When the next Python program (simple_example_write1_h5py.py) is run from the command line (and there are no problems), the simple_example_write1_h5py.hdf5 file is generated.

```python
#!/usr/bin/env python

"""
Writes the simplest NeXus HDF5 file using h5py

(continues on next page)
```
Uses method accepted at 2014NIAC
according to the example from Figure 1.3
in the Introduction chapter

```python
from pathlib import Path
import numpy

from nexusformat.nexus import NXdata, NXentry, NXfield, nxopen

filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
buffer = numpy.loadtxt(filename).T

tthData = buffer[0]
countsData = numpy.asarray(buffer[1], "int32")

with nxopen("simple_example_write1.hdf5", "w") as f:
    f["Scan"] = NXentry()
    tth = NXfield(tthData, name="two_theta", units="degrees")
    counts = NXfield(countsData, name="counts", units="counts")
    f["Scan/data"] = NXdata(counts, tth)
```

(continues on previous page)

h5py

#!/usr/bin/env python

"""
Writes the simplest NeXus HDF5 file using h5py

Uses method accepted at 2014NIAC
(continues on next page)
according to the example from Figure 1.3
in the Introduction chapter

```python
from pathlib import Path
import h5py
import numpy

filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
buffer = numpy.loadtxt(filename).T
tthData = buffer[0]  # float[]
countsData = numpy.asarray(buffer[1], "int32")  # int[]

with h5py.File("simple_example_write1.hdf5", "w") as f:
    nxentry = f.create_group("Scan")
    nxentry.attrs["NX_class"] = "NXentry"

    nxdata = nxentry.create_group("data")
    nxdata.attrs["NX_class"] = "NXdata"
    nxdata.attrs["signal"] = "counts"
    nxdata.attrs["axes"] = "two_theta"
    nxdata.attrs["two_theta_indices"] = [0,
    ]
    tth = nxdata.create_dataset("two_theta", data=tthData)
    tth.attrs["units"] = "degrees"

    counts = nxdata.create_dataset("counts", data=countsData)
    counts.attrs["units"] = "counts"
```

One of the tools provided with the HDF5 support libraries is the h5dump command, a command-line tool to print out the contents of an HDF5 data file. With no better tool in place (the output is verbose), this is a good tool to investigate what has been written to the HDF5 file. View this output from the command line using h5dump `simple_example_write1.hdf5`. Compare the data contents with the numbers shown above. Note that the various HDF5 data types have all been decided by the h5py support package.

**Note:** The only difference between this file and one written using the NAPI is that the NAPI file will have some additional, optional attributes set at the root level of the file that tells the original file name, time it was written, and some version information about the software involved.

Since the output of h5dump is verbose (see the Downloads section below), the punx tree tool¹ was used to print out the structure of HDF5 data files. This tool provides a simplified view of the NeXus file. Here is the output:

```text
Scan:NXentry
    @NX_class = "NXentry"
    data:NXdata
        @NX_class = "NXdata"
```

¹ punx tree: https://punx.readthedocs.io/en/latest/source_code/h5tree.html#how-to-use-h5tree
As the data files in these examples become more complex, you will appreciate the information density provided by *punx tree*.

**downloads**

The Python code and files related to this section may be downloaded from the following table.

<table>
<thead>
<tr>
<th>file</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>../simple_example.dat</td>
<td>2-column ASCII data used in this section</td>
</tr>
<tr>
<td>simple_example_write1.py</td>
<td>h5py code to write example <em>simple_example_write1</em></td>
</tr>
<tr>
<td>nexusformat/simple_example_write1.py</td>
<td>nexusformat code to write example <em>simple_example_write1</em></td>
</tr>
<tr>
<td>simple_example_write1.hdf5</td>
<td>NeXus file written by this code</td>
</tr>
<tr>
<td>simple_example_write1_h5dump.txt</td>
<td>h5dump analysis of the NeXus file</td>
</tr>
<tr>
<td>simple_example_write1_structure.txt</td>
<td><em>punx tree</em> analysis of the NeXus file</td>
</tr>
</tbody>
</table>

**Write a NeXus HDF5 file with plottable data**

Building on the previous example, we wish to identify our measured data with the detector on the instrument where it was generated. In this hypothetical case, since the detector was positioned at some angle `two_theta`, we choose to store both datasets, `two_theta` and `counts`, in a NeXus group. One appropriate NeXus group is `NXdetector`. This group is placed in a `NXinstrument` group which is placed in a `NXentry` group. To support a default plot, we provide a `NXdata` group. Rather than duplicate the same data already placed in the detector group, we choose to link to those datasets from the `NXdata` group. (Compare the next figure with Linking in a NeXus file in the NeXus Design chapter of the NeXus User Manual.) The NeXus Design chapter provides a figure (Linking in a NeXus file) with a small variation from our previous example, placing the measured data within the `/entry/instrument/detector` group. Links are made from that data to the `/entry/data` group.
The Python code to build an HDF5 data file with that structure (using numerical data from the previous example) is shown below.

```python
#!/usr/bin/env python

# Writes a simple NeXus HDF5 file using h5py with links
# according to the example from Figure 2.1 in the Design chapter

from pathlib import Path

import numpy

from nexusformat.nexus import (NXdata, NXdetector, NXentry, NXfield,
                               NXinstrument, NXlink, nxopen)

filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
buffer = numpy.loadtxt(filename).T
tthData = buffer[0]  # float[]
countsData = numpy.asarray(buffer[1], "int32")  # int[]

with nxopen("simple_example_write2.hdf5", "w") as f:
    # create the HDF5 NeXus file
    f["entry"] = NXentry()
    f["entry/instrument"] = NXinstrument()
    f["entry/instrument/detector"] = NXdetector()

    # store the data in the NXdetector group
    f["entry/instrument/detector/two_theta"] = NXfield(tthData, units="degrees")
    f["entry/instrument/detector/counts"] = NXfield(countsData, units="counts")
    f["entry/data"] = NXdata(NXlink("/entry/instrument/detector/counts"),
                              NXlink("/entry/instrument/detector/two_theta"))
    f["entry/data"].set_default()
```

```
# writes a simple NeXus HDF5 file using h5py with links
# according to the example from Figure 2.1 in the Design chapter

from pathlib import Path

import h5py
import numpy

filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
buffer = numpy.loadtxt(filename).T
tthData = buffer[0]  # float[]
countsData = numpy.asarray(buffer[1], "int32")  # int[]

with h5py.File("simple_example_write2.hdf5", "w") as f:
    f["entry"] = NXEntry()
    f["entry/instrument"] = NXInstrument()
    f["entry/instrument/detector"] = NXDetector()

    # store the data in the NXdetector group
    f["entry/instrument/detector/two_theta"] = NXField(tthData, units="degrees")
    f["entry/instrument/detector/counts"] = NXField(countsData, units="counts")
    f["entry/data"] = NXData(NXLink("/entry/instrument/detector/counts"),
                              NXLink("/entry/instrument/detector/two_theta"))
    f["entry/data"].set_default()
```

(continues on next page)
with h5py.File("simple_example_write2.hdf5", "w") as f:
    # create the HDF5 file
    f.attrs["default"] = "entry"

    nxentry = f.create_group("entry")
    nxentry.attrs["NX_class"] = "NXentry"
    nxentry.attrs["default"] = "data"

    nxinstrument = nxentry.create_group("instrument")
    nxinstrument.attrs["NX_class"] = "NXinstrument"

    nxdetector = nxinstrument.create_group("detector")
    nxdetector.attrs["NX_class"] = "NXdetector"

    # store the data in the NXdetector group
    ds_tth = nxdetector.create_dataset("two_theta", data=tthData)
    ds_tth.attrs["units"] = "degrees"
    ds_counts = nxdetector.create_dataset("counts", data=countsData)
    ds_counts.attrs["units"] = "counts"

    # create the NXdata group to define the default plot
    nxdata = nxentry.create_group("data")
    nxdata.attrs["NX_class"] = "NXdata"
    nxdata.attrs["signal"] = "counts"
    nxdata.attrs["axes"] = "two_theta"
    nxdata.attrs["two_theta_indices"] = [0,]

    source_addr = "/entry/instrument/detector/two_theta"  # existing data
    target_addr = "two_theta"  # new location
    ds_tth.attrs["target"] = source_addr  # a NeXus API convention for links
    nxdata[target_addr] = f[source_addr]  # hard link
    # nxdata._id.link(source_addr, target_addr, h5py.h5g.LINK_HARD)

    source_addr = "/entry/instrument/detector/counts"  # existing data
    target_addr = "counts"  # new location
    ds_counts.attrs["target"] = source_addr  # a NeXus API convention for links
    nxdata[target_addr] = f[source_addr]  # hard link
    # nxdata._id.link(source_addr, target_addr, h5py.h5g.LINK_HARD)

It is interesting to compare the output of the h5dump of the data file simple_example_write2.hdf5 with our Python instructions. See the downloads section below.

Look carefully! It appears in the output of h5dump that the actual data for two_theta and counts has moved into the NXdata group at HDF5 path /entry/data! But we stored that data in the NXdetector group at /entry/instrument/detector. This is normal for h5dump output.

A bit of explanation is necessary at this point. The data is not stored in either HDF5 group directly. Instead, HDF5 creates a DATA storage element in the file and posts a reference to that DATA storage element as needed. An HDF5 hard link requests another reference to that same DATA storage element. The h5dump tool describes in full that DATA storage element the first time (alphabetically) it is called. In our case, that is within the NXdata group. The next time it is called, within the NXdetector group, h5dump reports that a hard link has been made and shows the HDF5 path to the
NeXus recognizes this behavior of the HDF5 library and adds an additional structure when building hard links, the target attribute, to preserve the original location of the data. Not that it actually matters. the punx tree tool knows about the additional NeXus target attribute and shows the data to appear in its original location, in the NXdetector group.

```python
@default = "entry"
entry:NXentry
    @NX_class = "NXentry"
    @default = "data"
data:NXdata
    @NX_class = "NXdata"
    @axes = "two_theta"
    @signal = "counts"
    @two_theta_indices = [0]
counts --> /entry/instrument/detector/counts
two_theta --> /entry/instrument/detector/two_theta
instrument:NXinstrument
    @NX_class = "NXinstrument"
detector:NXdetector
    @NX_class = "NXdetector"
counts:NX_INT32[31] = [1037, 1318, 1704, '...', 1321]
    @target = "/entry/instrument/detector/counts"
    @units = "counts"
two_theta:NX_FLOAT64[31] = [17.92608, 17.92591, 17.92575, '...', 17.92108]
    @target = "/entry/instrument/detector/two_theta"
    @units = "degrees"
```

downloads

The Python code and files related to this section may be downloaded from the following table.

<table>
<thead>
<tr>
<th>file</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>../simple_example.dat</td>
<td>2-column ASCII data used in this section</td>
</tr>
<tr>
<td>simple_example_write2.py</td>
<td>h5py code to write example simple_example_write2</td>
</tr>
<tr>
<td>nexusformat/simple_example_write2.py</td>
<td>nexusformat code to write example simple_example_write2</td>
</tr>
<tr>
<td>simple_example_write2.hdf5</td>
<td>NeXus file written by this code</td>
</tr>
<tr>
<td>simple_example_write2_h5dump.txt</td>
<td>h5dump analysis of the NeXus file</td>
</tr>
<tr>
<td>simple_example_write2_structure.txt</td>
<td>punx tree analysis of the NeXus file</td>
</tr>
</tbody>
</table>

Write a NeXuS HDF5 File with links to external data

HDF5 files may contain links to data (or groups) in other files. This can be used to advantage to refer to data in existing HDF5 files and create NeXus-compliant data files. Here, we show such an example, using the same counts v. two_theta data from the examples above.

We use the HDF5 external file links with NeXus data files.

```python
f[local_addr] = h5py.ExternalLink(external_file_name, external_addr)
```
where \( f \) is an open \texttt{h5py.File()} object in which we will create the new link, \texttt{local_addr} is an HDF5 path address, \texttt{external_file_name} is the name (relative or absolute) of an existing HDF5 file, and \texttt{external_addr} is the HDF5 path address of the existing data in the \texttt{external_file_name} to be linked.

**file: external_angles.hdf5**

Take for example, the structure of \texttt{external_angles.hdf5}, a simple HDF5 data file that contains just the \texttt{two_theta} angles in an HDF5 dataset at the root level of the file. Although this is a valid HDF5 data file, it is not a valid NeXus data file:

1. \texttt{angles:float64[31] = [17.926079999999999, '\ldots', 17.92108]}
2. \texttt{@units = degrees}

**file: external_counts.hdf5**

The data in the file \texttt{external_angles.hdf5} might be referenced from another HDF5 file (such as \texttt{external_counts.hdf5}) by an HDF5 external link.\(^1\) Here is an example of the structure:

```
entry: NXentry
  instrument: NXinstrument
  detector: NXdetector
  counts: NX_INT32[31] = [1037, '\ldots', 1321]
    @units = counts
  two_theta --> file="external_angles.hdf5", path="/angles"
```

**file: external_master.hdf5**

A valid NeXus data file could be created that refers to the data in these files without making a copy of the data files themselves.

**Note:** It is necessary for all these files to be located together in the same directory for the HDF5 external file links to work properly.\(^1\)

To be a valid NeXus file, it must contain a \texttt{NXentry} group. For the files above, it is simple to make a master file that links to the data we desire, from structure that we create. We then add the group attributes that describe the default plottable data:

```
data: NXdata
  @signal = counts
  @axes = "two_theta"
  @two_theta_indices = []
```

Here is (the basic structure of) \texttt{external_master.hdf5}, an example:

```
entry: NXentry
  @default = data
  instrument --> file="external_counts.hdf5", path="/entry/instrument"
```

\(^1\) see these URLs for further guidance on HDF5 external links: https://portal.hdfgroup.org/display/HDF5/H5L_CREATE_EXTERNAL, http://docs.h5py.org/en/stable/high/group.html#external-links

---

98 Chapter 2. Examples of writing and reading NeXus data files
Here is the complete code of a Python program, using h5py to write a NeXus-compliant HDF5 file with links to data in other HDF5 files.

**external_example_write.py**: Write using HDF5 external links

```python
# Example code for writing a NeXus HDF5 file using h5py with links to data in other HDF5 files.
# This example is based on `writer_2_1`.

from pathlib import Path
import h5py
import numpy
from nexusformat.nexus import (NXdata, NXdetector, NXentry, NXfield,
                               NXinstrument, NXlink, nxopen)

FILE_HDF5_MASTER = "external_master.hdf5"
FILE_HDF5_ANGLES = "external_angles.hdf5"
FILE_HDF5_COUNTS = "external_counts.hdf5"

# get some data
filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
buffer = numpy.loadtxt(filename).T
tthData = buffer[0]  # float[]
countsData = numpy.asarray(buffer[1], "int32")  # int[]

# put the angle data in an external (non-NeXus) HDF5 data file
with h5py.File(FILE_HDF5_ANGLES, "w") as f:
    ds = f.create_dataset("angles", data=tthData)
    ds.attrs["units"] = "degrees"

# put the detector counts in an external HDF5 data file
# with *incomplete* NeXus structure (no NXdata group)
```

(continues on next page)
with nxopen(FILE_HDF5_COUNTS, "w") as f:
    f["entry"] = NXentry()
    f["entry/instrument"] = NXinstrument()
    f["entry/instrument/detector"] = NXdetector()
    f["entry/instrument/detector/counts"] = NXfield(countsData, units="counts")
    f["entry/instrument/detector/two_theta"] = NXlink("/angles", FILE_HDF5_ANGLES)

# create a master NeXus HDF5 file
with nxopen(FILE_HDF5_MASTER, "w") as f:
    f["entry"] = NXentry()
    counts = NXlink("/entry/instrument/detector/counts", FILE_HDF5_COUNTS, name="counts")
    two_theta = NXlink("/angles", FILE_HDF5_ANGLES, name="two_theta")
    f["entry/data"] = NXdata(counts, two_theta)
    f["entry/data"].set_default()
    f["entry/instrument"] = NXlink("/entry/instrument", FILE_HDF5_COUNTS)

h5py

#!/usr/bin/env python

"""
Writes a NeXus HDF5 file using h5py with links to data in other HDF5 files.

This example is based on "writer_2_1".
"""

from pathlib import Path
import h5py
import numpy

FILE_HDF5_MASTER = "external_master.hdf5"
FILE_HDF5_ANGLES = "external_angles.hdf5"
FILE_HDF5_COUNTS = "external_counts.hdf5"

# ---------------------------

# get some data
filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
buffer = numpy.loadtxt(filename).T
tthData = buffer[0]  # float]
countsData = numpy.asarray(buffer[1], "int32")  # int[]

# put the angle data in an external (non-NeXus) HDF5 data file
with h5py.File(FILE_HDF5_ANGLES, "w") as f:
    ds = f.create_dataset("angles", data=tthData)
    ds.attrs["units"] = "degrees"

# put the detector counts in an external HDF5 data file
# with "incomplete" NeXus structure (no NXdata group)
with h5py.File(FILE_HDF5_COUNTS, "w") as f:
    nxentry = f.create_group("entry")
nxentry.attrs["NX_class"] = "NXEntry"
nxinstrument = nxentry.create_group("instrument")
nxinstrument.attrs["NX_class"] = "NXInstrument"
nxdetector = nxinstrument.create_group("detector")
nxdetector.attrs["NX_class"] = "NXDetector"
data = nxdetector.create_dataset("counts", data=countsData)
data.attrs["units"] = "counts"

# link the "two_theta" data stored in separate file
local_addr = nxdetector.name + "/two_theta"

with h5py.File(FILE_HDF5_MASTER, "w") as f:
    f.attrs["default"] = "entry"
    nxentry = f.create_group("entry")
    nxentry.attrs["NX_class"] = "NXEntry"
    nxentry.attrs["default"] = "data"
    nxdata = nxentry.create_group("data")
    nxdata.attrs["NX_class"] = "NXData"
    nxdata.attrs["signal"] = "counts"
    nxdata.attrs["axes"] = "two_theta"
    nxdata.attrs["two_theta_indices"] = [
        0,
    ]

    local_addr = "/entry/instrument"
    f[local_addr] = h5py.ExternalLink(FILE_HDF5_COUNTS, "/entry/instrument")

 downloads

The Python code and files related to this section may be downloaded from the following table.
Find plottable data in a NeXus HDF5 file

Let’s make a new reader that follows the chain of attributes (@default, @signal, and @axes) to find the default plottable data. We’ll use the same data file as the previous example. Our demo here assumes one-dimensional data. (For higher dimensionality data, we’ll need more complexity when handling the @axes attribute and we’ll to check the field sizes. See section Find the plottable data, subsection Version 3, for the details.)

reader_attributes_trail.py: Read a NeXus HDF5 file using Python

```python
from pathlib import Path
from nexusformat.nexus import nxopen

filename = str(
    Path(__file__).absolute().parent.parent
    / "simple_example_basic"
    / "simple_example_basic.nexus.hdf5"
)

with nxopen(filename) as f:
    # find the default NXdata group
    nx_data = f.get_default()
    signal = nx_data.nxsignal
    axes = nx_data.nxaxes[0]

nx_data.plot() # plot the data using Matplotlib

print(f"file: {f.nxfilename}")
print(f"signal: {signal.nxname}")
print(f"axes: {axes.nxname}")
print(f\"{axes.nxname} \{signal.nxname}\")
for x, y in zip(axes, signal):
    print(x, y)
```

h5py

```python
from pathlib import Path
import h5py
```

(continues on next page)
filename = str(
    Path(__file__).absolute().parent.parent
    /"simple_example_basic"
    /"simple_example_basic.nexus.hdf5"
)

with h5py.File(filename, "r") as nx:
    # find the default NXentry group
    nx_entry = nx[nx.attrs["default"]]
    # find the default NXdata group
    nx_data = nx_entry[nx_entry.attrs["default"]]
    # find the signal field
    signal = nx_data[nx_data.attrs["signal"]]
    # find the axes field(s)
    attr_axes = nx_data.attrs["axes"]
    if isinstance(attr_axes, (set, tuple, list)):
        # but check that attr_axes only describes 1-D data
        if len(attr_axes) == 1:
            attr_axes = attr_axes[0]
        else:
            raise ValueError(f"expected 1-D data but @axes={attr_axes}"
    axes = nx_data[attr_axes]

print(f"file: {nx.filename}")
print(f"signal: {signal.name}")
print(f"axes: {axes.name}")
for x, y in zip(axes, signal):
    print(x, y)
17.92391 66863
17.92375 66599
17.92358 66206
17.92341 65747
17.92325 65250
17.92308 64129
17.92275 60796
17.92258 56795
17.92241 51550
17.92225 43710
17.92208 29315
17.92191 19782
17.92175 12992
17.92158 6622
17.92141 4198
17.92125 2248
17.92108 1321

downloads

The Python code and files related to this section may be downloaded from the following table.

<table>
<thead>
<tr>
<th>file</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reader_attributes_trail.py</td>
<td>h5py code to read NeXus HDF5 file and find plottable data</td>
</tr>
<tr>
<td>nexusformat/reader_attributes_trail.py</td>
<td>nexusformat code to read NeXus HDF5 file and find plottable data</td>
</tr>
</tbody>
</table>

- Write examples with nexusformat for different NeXus classes
- Write examples with h5py for different NeXus classes

Example data used

The data shown plotted in the next figure will be written to the NeXus HDF5 file using only two NeXus base classes, NXentry and NXdata, in the first example and then minor variations on this structure in the next two examples. The data model is identical to the one in the Introduction chapter except that the names will be different, as shown below:

Fig. 3: data structure of the simple example
Simple example values

<table>
<thead>
<tr>
<th>nr</th>
<th>( \text{I00} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1721</td>
<td>1037</td>
</tr>
<tr>
<td>0.1725</td>
<td>1318</td>
</tr>
<tr>
<td>0.1727</td>
<td>1704</td>
</tr>
<tr>
<td>0.1728</td>
<td>2857</td>
</tr>
<tr>
<td>0.1729</td>
<td>4516</td>
</tr>
<tr>
<td>0.1729</td>
<td>9998</td>
</tr>
<tr>
<td>0.1730</td>
<td>23819</td>
</tr>
<tr>
<td>0.1730</td>
<td>31662</td>
</tr>
<tr>
<td>0.1731</td>
<td>40458</td>
</tr>
<tr>
<td>0.1732</td>
<td>49087</td>
</tr>
<tr>
<td>0.1732</td>
<td>56514</td>
</tr>
<tr>
<td>0.1733</td>
<td>63499</td>
</tr>
<tr>
<td>0.1733</td>
<td>66802</td>
</tr>
<tr>
<td>0.1734</td>
<td>66863</td>
</tr>
<tr>
<td>0.1734</td>
<td>66599</td>
</tr>
<tr>
<td>0.1735</td>
<td>66206</td>
</tr>
<tr>
<td>0.1735</td>
<td>65747</td>
</tr>
<tr>
<td>0.1736</td>
<td>65250</td>
</tr>
<tr>
<td>0.1736</td>
<td>64129</td>
</tr>
<tr>
<td>0.1737</td>
<td>63044</td>
</tr>
<tr>
<td>0.1737</td>
<td>60796</td>
</tr>
</tbody>
</table>

Fig. 4: plot of the simple example data

```
/entry:NXentry
 /mr_scan:NXdata
   /mr : float64[31]
   /I00 : int32[31]
```

(continues on next page)
2.1.3 HDF5 in MATLAB

author
Paul Kienzle, NIST

Note: Editor's Note: These files were copied directly from an older version of the NeXus documentation (DocBook) and have not been checked that they will run under current Matlab versions.

input.dat

This is the same data used with HDF5 in Python.
writing data

*basic_writer.m: Write a NeXus HDF5 file using Matlab*

```matlab
% Writes a NeXus HDF5 file using matlab

filename = 'prj_test.nexus.hdf5';
timestamp = '2010-10-18T17:17:04-0500';

% read input data
A = load('input.dat');
mr = A(:,1);
I00 = int32(A(:,2));

% clear out old file, if it exists
delete(filename);

% using the simple h5 interface, there is no way to create a group without
% first creating a dataset; creating the dataset creates all intervening
% groups.

% store x
h5create(filename,'/entry/mr_scan/mr',[length(mr)]);
h5write(filename,'/entry/mr_scan/mr',mr);
h5writeatt(filename,'/entry/mr_scan/mr','units','degrees');
h5writeatt(filename,'/entry/mr_scan/mr','long_name','USAXS mr (degrees)');

% store y
h5create(filename,'/entry/mr_scan/I00',[length(I00)],'DataType','int32');
h5write(filename,'/entry/mr_scan/I00',I00);
h5writeatt(filename,'/entry/mr_scan/I00','units','counts');
h5writeatt(filename,'/entry/mr_scan/I00','long_name','USAXS I00 (counts)');

% indicate that we are plotting y vs. x
h5writeatt(filename,'/','default','entry');
h5writeatt(filename,'/','default','mr_scan');
h5writeatt(filename,'/','default','signal',I00);
h5writeatt(filename,'/','default','axes',mr_scan);
h5writeatt(filename,'/','default','mr_scan_indices', int32(0));

% add NeXus metadata

(continues on next page)
```
reading data

basic_reader.m: Read a NeXus HDF5 file using Matlab

% Reads NeXus HDF5 file and print the contents
1            filename = 'prj_test.nexus.hdf5';
2            root = h5info(filename,'/');
3            attrs = root.Attributes;
4            for i = 1:length(attrs)
5                fprintf('%s: %s\n', attrs(i).Name, attrs(i).Value);
6            end
7            mr = h5read(filename,'/entry/mr_scan/mr');
8            I00 = h5read(filename,'/entry/mr_scan/I00');
9            fprintf('#\t%gs\t%gs\n', 'mr', 'I00');
10           for i = 1:length(mr)
11                fprintf('%d\t%g\t%d\n', i, mr(i), I00(i));
12           end

writing data file with links

writer_2_1.m: Write a NeXus HDF5 file with links

% Writes a simple NeXus HDF5 file with links
% according to the example from Figure 2.1 in the Design chapter
1            filename = 'writer_2_1.hdf5';
2            % read input data
3            A = load('input.dat');
4            two_theta = A(:,1);
5            counts = int32(A(:,2));
6            % clear out old file, if it exists
7            delete(filename);
8            % store x
h5create(filename,'/entry/instrument/detector/two_theta',[length(two_theta)]);
15 h5write(filename,'/entry/instrument/detector/two_theta',two_theta);
16 h5writeatt(filename,'/entry/instrument/detector/two_theta','units','degrees');

% store y
h5create(filename,'/entry/instrument/detector/counts',[length(counts)],'DataType','int32');
19 h5write(filename,'/entry/instrument/detector/counts',counts);
20 h5writeatt(filename,'/entry/instrument/detector/counts','units','counts');

% create group NXdata with links to detector
% note: requires the additional file h5link.m
h5link(filename,'/entry/instrument/detector/two_theta','/entry/data/two_theta');
25 h5link(filename,'/entry/instrument/detector/counts','/entry/data/counts');

% indicate that we are plotting y vs. x
h5writeatt(filename,'/entry','default','entry');
29 h5writeatt(filename,'/entry','default','data');
30 h5writeatt(filename,'/entry/data','signal','counts');
31 h5writeatt(filename,'/entry/data','axes','two_theta');
32 h5writeatt(filename,'/entry/data','two_theta_indices','int32(0)');

% add NeXus metadata
h5writeatt(filename,'/','file_name',filename);
36 h5writeatt(filename,'/','file_time',timestamp);
37 h5writeatt(filename,'/','instrument','APS USAXS at 32ID-B');
38 h5writeatt(filename,'/','creator','writer_2_1.m');
39 h5writeatt(filename,'/','NeXus_version','4.3.0');
40 h5writeatt(filename,'/','HDF5_Version','1.6'); % no 1.8 features used in this example
41 h5writeatt(filename,'/','NX_class','NXentry');
42 h5writeatt(filename,'/','NX_class','NXinstrument');
43 h5writeatt(filename,'/','NX_class','NXdetector');
44 h5writeatt(filename,'/','NX_class','NXdata');

% show structure of the file that was created
h5disp(filename);

\textit{h5link.m: support module for creating NeXus-style HDF5 hard links}

function h5link(filename, from, to)
%H5LINK Create link to an HDF5 dataset.
% H5LINK(FILENAME,SOURCE,TARGET) creates an HDF5 link from the
% dataset at location SOURCE to a dataset at location TARGET. All
% intermediate groups in the path to target are created.

% Example: create a link from /hello/world to /goodbye/world
h5create('myfile.h5','/hello/world',[100 200]);
% h5link('myfile.h5','/hello/world','/goodbye/world');
% hgdisp('myfile.h5');
% See also: h5create, h5read, h5write, h5info, h5disp

% split from and to into group/dataset
idx = strfind(from,'/');
from_path = from(1:idx(end)-1);
from_data = from(idx(end)+1:end);
idx = strfind(to,'/');
to_path = to(1:idx(end)-1);
to_data = to(idx(end)+1:end);

% open the HDF file
fid = H5F.open(filename,'H5F_ACC_RDWR','H5P_DEFAULT');

% create target group if it doesn't already exist
create_intermediate = H5P.create('H5P_LINK_CREATE');
H5P.set_create_intermediate_group(create_intermediate, 1);
try
    H5G.create(fid,to_path,create_intermediate,'H5P_DEFAULT','H5P_DEFAULT');
catch
end
H5P.close(create_intermediate);

% open groups and create link
from_id = H5G.open(fid, from_path);
to_id = H5G.open(fid, to_path);
H5L.create_hard(from_id, from_data, to_id, to_data, 'H5P_DEFAULT','H5P_DEFAULT');

% close all
H5G.close(from_id);
H5G.close(to_id);
H5F.close(fid);
end

Downloads

<table>
<thead>
<tr>
<th>file</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input.dat</td>
<td>two-column text data file, also used in other examples</td>
</tr>
<tr>
<td>basic_writer.m</td>
<td>writes a NeXus HDF5 file using input.dat</td>
</tr>
<tr>
<td>basic_reader.m</td>
<td>reads the NeXus HDF5 file written by basic_writer.m</td>
</tr>
<tr>
<td>h5link.m</td>
<td>support module for creating NeXus-style HDF5 hard links</td>
</tr>
<tr>
<td>writer_2_1.m</td>
<td>like basic_writer.m but stores data in /entry/instrument/detector and then links to NXdata group</td>
</tr>
</tbody>
</table>
2.1.4 HDF5 in C with NAPI

Code examples are provided in this section that write 2-D data to a NeXus HDF5 file in the C language using the NAPI: NeXus Application Programmer Interface (frozen).

The following code reads a two-dimensional set \textit{counts} with dimension scales of \textit{t} and \textit{phi} using local routines, and then writes a NeXus file containing a single \texttt{NXentry} group and a single \texttt{NXdata} group. This is the simplest data file that conforms to the NeXus standard.

**NAPI C Example: write simple NeXus file**

Note: This example uses the signal/axes attributes applied to the data field, as described in \textit{Associating plottable data by name using the axes attribute}. New code should use the method described in \textit{Associating plottable data using attributes applied to the NXdata group}.

```c
#include "napi.h"

int main()
{
  int counts[50][1000], n_t=1000, n_p=50, dims[2], i;
  float t[1000], phi[50];
  NXhandle file_id;

  /* Read in data using local routines to populate phi and counts
   * for example you may create a getdata() function and call
   *  getdata (n_t, t, n_p, phi, counts);
   */

  /* Open output file and output global attributes */
  NXopen ("NXfile.nxs", NXACC_CREATE5, &file_id);
  NXputattr (file_id, "user_name", "Joe Bloggs", 10, NX_CHAR);

  /* Open top-level NXentry group */
  NXmakegroup (file_id, "Entry1", "NXentry");
  NXopengroup (file_id, "Entry1", "NXentry");

  /* Open NXdata group within NXentry group */
  NXmakegroup (file_id, "Data1", "NXdata");
  NXopengroup (file_id, "Data1", "NXdata");

  /* Output time channels */
  NXmakedata (file_id, "time_of_flight", NX_FLOAT32, 1, &n_t);
  NXopendata (file_id, "time_of_flight");
  NXputdata (file_id, t);
  NXputattr (file_id, "units", "microseconds", 12, NX_CHAR);
  NClosedata (file_id);

  /* Output detector angles */
  NXmakedata (file_id, "polar_angle", NX_FLOAT32, 1, &n_p);
  NXopendata (file_id, "polar_angle");
  NXputdata (file_id, phi);
  NXputattr (file_id, "units", "degrees", 7, NX_CHAR);
  NClosedata (file_id);

  /* Output data */
```

(continues on next page)
2.1.5 HDF5 in Fortran with NAPI

Code examples are provided in this section that write 2-D data to a NeXus HDF5 file in F77, and F90 languages using the NAPI: NeXus Application Programmer Interface (frozen).

The following code reads a two-dimensional set counts with dimension scales of \( t \) and \( \phi \) using local routines, and then writes a NeXus file containing a single NXentry group and a single NXdata group. This is the simplest data file that conforms to the NeXus standard.

NAPI F77 Example: write simple NeXus file

Note: The F77 interface is no longer being developed.
NAPI F90 Example: write simple NeXus file

Note: This example uses the signal/axes attributes applied to the data field, as described in Associating plottable data by name using the axes attribute. New code should use the method described in Associating plottable data using attributes applied to the NXdata group.

```fortran
program WRITEDATA
  use NXUmodule

  type(NXhandle) :: file_id
  integer, pointer :: counts(:,:)
  real, pointer :: t(:), phi(:)

  !Use local routines to allocate pointers and fill in data
  call getlocaldata (t, phi, counts)

  !Open output file
  if (NXopen ("NXfile.nxs", NXACC_CREATE, file_id) /= NX_OK) stop
  if (NXUwriteglobals (file_id, user="Joe Bloggs") /= NX_OK) stop

  !Output detector angles
  dims(1) = n_t
  dims(2) = n_p
  status = NXmakedata (file_id, 'polar_angle', NX_FLOAT32, 1, dims)
  status = NXopendata (file_id, 'polar_angle')
  status = NXputdata (file_id, phi)
  status = NXputcharattr (file_id, 'units', 'degrees', 7, NX_CHAR)
  status = NXclosedata (file_id)

  !Output data
  status = NXmakedata (file_id, 'counts', NX_INT32, 2, dims)
  status = NXopendata (file_id, 'counts')
  status = NXputdata (file_id, counts)
  status = NXputattr (file_id, 'signal', 1, 1, NX_INT32)
  status = NXputattr (file_id, 'axes', 'polar_angle:time_of_flight', 26, NX_CHAR)
  status = NXclosedata (file_id)

  !Close NXdata and NXentry groups and close file
  status = NXclosegroup (file_id)
  status = NXclosegroup (file_id)
  status = NXclose (file_id)

  stop
end
```
!Set compression parameters
if (NXUsetcompress (file_id, NX_COMP_LZW, 1000) /= NX_OK) stop

!Open top-level NXentry group
if (NXUwritegroup (file_id, "Entry1", "NXentry") /= NX_OK) stop

!Open NXdata group within NXentry group
if (NXUwritegroup (file_id, "Data1", "NXdata") /= NX_OK) stop

!Output time channels
if (NXUwritedata (file_id, "time_of_flight", t, "microseconds") /= NX_OK) stop

!Output detector angles
if (NXUwritedata (file_id, "polar_angle", phi, "degrees") /= NX_OK) stop

!Output data
if (NXUwritedata (file_id, "counts", counts, "counts") /= NX_OK) stop
if (NXputattr (file_id, "signal", 1) /= NX_OK) stop
if (NXputattr (file_id, "axes", "polar_angle:time_of_flight") /= NX_OK) stop

!Close NXdata group
if (NXclosegroup (file_id) /= NX_OK) stop

!Close NXentry group
if (NXclosegroup (file_id) /= NX_OK) stop

!Close NeXus file
if (NXclose (file_id) /= NX_OK) stop

end program WRITEDATA


2.1.6 HDF5 in Python with NAPI

A single code example is provided in this section that writes 3-D data to a NeXus HDF5 file in the Python language using the NAPI: NeXus Application Programmer Interface (frozen).

The data to be written to the file is a simple three-dimensional array (2 x 3 x 4) of integers. The single dataset is intended to demonstrate the order in which each value of the array is stored in a NeXus HDF5 data file.

NAPI Python Example: write simple NeXus file

#!/usr/bin/python
import sys
import nxs
import numpy

a = numpy.zeros((2,3,4),dtype=numpy.int)
val = 0
for i in range(2):
    for j in range(3):
        for k in range(4):
            a[i,j,k] = val
            val = val + 1

nf = nxs.open("simple3D.h5", "w5")
nf.makegroup("entry","NXentry")
2.2 Visualization tools

Tools to visualize NeXus HDF5 files graphically or in text form.

2.2.1 Plot a NeXus HDF5 file with NeXpy

A NeXus HDF5 file with plottable data (see Find plottable data in a NeXus HDF5 file) can be plotted by NeXpy\(^1\).

Compare this with plot of the simple example data and note that the horizontal axis of this plot is mirrored from that above. This is because the data is stored in the file in descending order and NeXpy has plotted it that way (in order of appearance) by default.

2.2.2 Plot a NeXus HDF5 file with silx view

A NeXus HDF5 file with plottable data (see Find plottable data in a NeXus HDF5 file) can be plotted by the silx view\(^1\) tool provided as part of silx\(^2\).

2.2.3 View a NeXus HDF5 file with punx tree

The punx tree tool\(^1\) provided as part of punx\(^2\) can be used to print the content of an HDF5 file. As an example we show the result of the command punx tree simple3D.h5 on the result of HDF5 in Python with NAPI
Fig. 5: plot the simple example using NeXpy

Fig. 6: plot the simple example using silx
2.2.4 View a NeXus HDF5 file with \textit{h5dump}

The \textit{h5dump} tool\footnote{h5dump: https://support.hdfgroup.org/HDF5/doc/RM/Tools.html#Tools-Dump} provided as part of the HDF5 tool kit\footnote{HDF5 tools: https://support.hdfgroup.org/products/hdf5_tools/} can be used to print the content of an HDF5 file. As an example we show the result of the command \texttt{h5dump simple3D.h5} on the result of \textit{HDF5 in Python with NAPI}

\begin{verbatim}
HDF5 "simple3D.h5" {
    GROUP "/" {
        ATTRIBUTE "NeXus_version" {
            DATATYPE H5T_STRING {
                STRSIZE 5;
                STRPAD H5T_STR_NULLTERM;
                CSET H5T_CSET_ASCII;
                CTYPE H5T_C_S1;
            }
            DATASPACE SCALAR
            DATA {
                (0): "4.1.0"
            }
        }
        ATTRIBUTE "file_name" {
            DATATYPE H5T_STRING {
                STRSIZE 11;
                STRPAD H5T_STR_NULLTERM;
                CSET H5T_CSET_ASCII;
                CTYPE H5T_C_S1;
            }
            DATASPACE SCALAR
        }
    }
}
\end{verbatim}
DATA {
  (0): "simple3D.h5"
}

ATTRIBUTE "HDF5_Version" {
  DATATYPE H5T_STRING {
    STRSIZE 5;
    STRPAD H5T_STR_NULLTERM;
    CSET H5T_CSET_ASCII;
    CTYPE H5T_C_S1;
  }
  DATASPACE SCALAR
  DATA {
    (0): "1.6.6"
  }
}

ATTRIBUTE "file_time" {
  DATATYPE H5T_STRING {
    STRSIZE 24;
    STRPAD H5T_STR_NULLTERM;
    CSET H5T_CSET_ASCII;
    CTYPE H5T_C_S1;
  }
  DATASPACE SCALAR
  DATA {
    (0): "2011-11-18 17:26:27+0100"
  }
}

GROUP "entry" {
  ATTRIBUTE "NX_class" {
    DATATYPE H5T_STRING {
      STRSIZE 7;
      STRPAD H5T_STR_NULLTERM;
      CSET H5T_CSET_ASCII;
      CTYPE H5T_C_S1;
    }
    DATASPACE SCALAR
    DATA {
      (0): "NXentry"
    }
  }
  GROUP "data" {
    ATTRIBUTE "NX_class" {
      DATATYPE H5T_STRING {
        STRSIZE 6;
        STRPAD H5T_STR_NULLTERM;
        CSET H5T_CSET_ASCII;
        CTYPE H5T_C_S1;
      }
      DATASPACE SCALAR
      DATA {
        (0): "NXdata"
      }
    }
  }
}
2.3 Examples for Specific Instruments

Examples of working with data from specific instruments.

2.3.1 Viewing 2-D Data from LRMECS

The IPNS LRMECS instrument stored data in NeXus HDF4 data files. One such example is available from the repository of NeXus data file examples. For this example, we will start with a conversion of that original data file into HDF5 format.

<table>
<thead>
<tr>
<th>format</th>
<th>file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDF4</td>
<td>lrc3701.nxs</td>
</tr>
<tr>
<td>HDF5</td>
<td>lrc3701.nx5</td>
</tr>
</tbody>
</table>

This dataset contains two histograms with 2-D images (148x750 and 148x32) of 32-bit integers. First, we use the h5dump tool to investigate the header content of the file (not showing any of the data).

---

1 LRMECS example data: https://github.com/nexusformat/exampledata/tree/master/IPNS/LRMECS
Visualize Using \texttt{h5dump}

Here, the output of the command:

\begin{verbatim}
\texttt{h5dump -H lrcs3701.nx5}
\end{verbatim}

has been edited to only show the first \texttt{NXdata} group (/Histogram1/data):

\begin{verbatim}
HDF5 "C:\Users\Pete\Documents\eclipse\NeXus\definitions\exampledata\IPNS\LRMECS\lrcs3701.nx5" {
  GROUP "/Histogram1/data" {
    DATASET "data" {
      DATATYPE H5T_STD_I32LE
      DATASPACE SIMPLE { ( 148, 750 ) / ( 148, 750 ) }
    }
    DATASET "polar_angle" {
      DATATYPE H5T_IEEE_F32LE
      DATASPACE SIMPLE { ( 148 ) / ( 148 ) }
    }
    DATASET "time_of_flight" {
      DATATYPE H5T_IEEE_F32LE
      DATASPACE SIMPLE { ( 751 ) / ( 751 ) }
    }
    DATASET "title" {
      DATATYPE H5T_STRING {
        STRSIZE 44;
        STRPAD H5T_STR_NULLTERM;
        CSET H5T_CSET_ASCII;
        CTYPE H5T_C_S1;
      }
      DATASPACE SIMPLE { ( 1 ) / ( 1 ) }
    }
  }
}
\end{verbatim}

Visualize Using \texttt{HDFview}

For many, the simplest way to view the data content of an HDF5 file is to use the \texttt{HDFview} program (https://portal.hdfgroup.org/display/HDFVIEW/HDFView) from The HDF Group. After starting \texttt{HDFview}, the data file may be loaded by dragging it into the main HDF window. On opening up to the first \texttt{NXdata} group /Histogram1/data (as above), and then double-clicking the dataset called: \texttt{data}, we get our first view of the data.

The data may be represented as an image by accessing the \textit{Open As} menu from HDFview (on Windows, right click the dataset called \texttt{data} and select the \textit{Open As} item, consult the HDFview documentation for different platform instructions). Be sure to select the \textit{Image} radio button, and then (accepting everything else as a default) press the \textit{Ok} button.

\textbf{Note:} In this image, dark represents low intensity while white represents high intensity.
2.3. Examples for Specific Instruments

Fig. 7: LRMECS lrcs3701 data: *HDFview*

Fig. 8: LRMECS lrcs3701 data: *HDFview Open As* dialog
Another way to visualize this data is to use a commercial package for scientific data visualization and analysis. One such package is IgorPro from http://www.wavemetrics.com

IgorPro provides a browser for HDF5 files that can open our NeXus HDF5 and display the image. Follow the instructions from WaveMetrics to install the HDF5 Browser package: http://www.wavemetrics.com/products/igorpro/dataaccess/hdf5.htm

You may not have to do this step if you have already installed the HDF5 Browser. IgorPro will tell you if it is not installed properly. To install the HDF5 Browser, first start IgorPro. Next, select from the menus and submenus: Data; Load Waves; Packages; Install HDF5 Package as shown in the next figure. IgorPro may direct you to perform more activities before you progress from this step.

Next, open the HDF5 Browser by selecting from the menus and submenus: Data; Load Waves; New HDF5 Browser as shown in the next figure.

Next, click the Open HDF5 File button and open the NeXus HDF5 file lrcs3701.nxs. In the lower left Groups panel, click the data dataset. Also, under the panel on the right called Load Dataset Options, choose No Table as shown. Finally, click the Load Dataset button (in the Datasets group) to display the image.

Note: In this image, dark represents low intensity while white represents high intensity. The image has been rotated for easier representation in this manual.
Fig. 10: LRMECS lrcs3701 data: *IgorPro* install HDF5 Browser

2.3. Examples for Specific Instruments
Fig. 11: LRMECS lrcs3701 data: IgorPro HDFBrowser dialog
Fig. 12: LRMECS lrcs3701 data: *IgorPro HDFBrowser* dialog

Fig. 13: LRMECS lrcs3701 data: *IgorPro* Image
2.3.2 EPICS Area Detector Examples

Two examples in this section show how to write NeXus HDF5 data files with EPICS Area Detector images. The first shows how to configure the HDF5 File Writing Plugin of the EPICS Area Detector software. The second example shows how to write an EPICS Area Detector image using Python.

HDF5 File Writing Plugin

This example describes how to write a NeXus HDF5 data file using the EPICS\(^1\) Area Detector\(^2\) HDF5 file writing plugin\(^3\). We will use the EPICS SimDetector\(^4\) as an example. (PV prefix: 13SIM1:) Remember to replace that with the prefix for your detector's IOC.

One data file will be produced for each image generated by EPICS.

You’ll need AreaDetector version 2.5 or higher to use this as the procedures for using the HDF5 file writing plugin changed with this release.

configuration files

There are two configuration files we must edit to configure an EPICS AreaDetector to write NeXus files using the HDF5 File Writer plugin:

<table>
<thead>
<tr>
<th>file</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes.xml</td>
<td>what information to know about from EPICS and other sources</td>
</tr>
<tr>
<td>layout.xml</td>
<td>where to write that information in the HDF5 file</td>
</tr>
</tbody>
</table>

Put these files into a known directory where your EPICS IOC can find them.

attributes.xml

The attributes file is easy to edit. Any text editor will do. A wide screen will be helpful.

Each `<Attribute />` element declares a single `ndattribute` which is associated with an area detector image. These `ndattribute` items can be written to specific locations in the HDF5 file or placed by default in a `default location`.

**Note:** The attributes file shown here has been reformatted for display in this manual. The `downloads` section below provides an attributes file with the same content using its wide formatting (one complete Attribute per line). Either version of this file is acceptable.

```xml
<?xml version="1.0" standalone="no" ?>
<!-- Attributes -->
<Attributes
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="https://github.com/areaDetector/ADCore/blob/master/iocBoot/NDAttributes.xsd"
>
```

1 EPICS: https://epics-controls.org/
2 EPICS Area Detector: https://areadetector.github.io/master/index.html
3 HDF5 File Writer: https://areadetector.github.io/master/ADCore/NDFileHDF5.html
4 EPICS SimDetector: https://github.com/areaDetector/ADSimDetector
If you want to add additional EPICS process variables (PVs) to be written in the HDF5 file, create additional \texttt{<Attribute />} elements (such as the \texttt{calc1\_val}) and modify the \texttt{name}, \texttt{source}, and \texttt{description} values. Be sure to use a unique \texttt{name} for each \texttt{ndattribute} in the attributes file.

\textbf{Note:} \texttt{ndattribute}: item specified by an \texttt{<Attribute />} element in the attributes file.
layout.xml

You might not need to edit the layout file. It will be fine (at least a good starting point) as it is, even if you add PVs (a.k.a. ndattribute) to the attributes.xml file.

```xml
<?xml version="1.0" standalone="no" ?>
<hdf5_layout>
  <group name="entry">
    <attribute name="NX_class" source="constant" value="NXentry" type="string"/>
    <group name="instrument">
      <attribute name="NX_class" source="constant" value="NXinstrument" type="string"/>
      <group name="detector">
        <dataset name="data" source="detector" det_default="true">
          <!-- The "target" attribute in /entry/instrument/detector/data is used to tell Nexus utilities that this is a hardlink -->
          <attribute name="NX_class" source="constant" value="NXdata" type="string"/>
          <hardlink name="data" target="/entry/instrument/detector/data"/>
        </dataset>
        <group name="NDAttributes ndattr_default="true">
          <attribute name="NX_class" source="constant" value="NXcollection" type="string"/>
        </group>
      </group>
    </group>
    <dataset name="ColorMode" source="ndattribute" ndattribute="ColorMode"/>
  </group>
  <group name="NDAttributes">
    <attribute name="NX_class" source="constant" value="NXcollection" type="string"/>
  </group>
  <!-- end group NDAttribute -->
  <group name="performance">
    <dataset name="timestamp" source="ndattribute" ndattribute="Timestamp"/>
  </group>
  <!-- end group performance -->
  <group name="data">
    <attribute name="NX_class" source="constant" value="NXdata" type="string"/>
    <hardlink name="data" target="/entry/instrument/detector/data"/>
  </group>
  <!-- end group data -->
  <group name="entry">
    <!-- end group entry -->
  </group>
</hdf5_layout>
```

If you do not specify where in the file to write an ndattribute from the attributes file, it will be written within the group that has ndattr_default="true". This identifies the group to the HDF5 file writing plugin as the default location to store content from the attributes file. In the example layout file, that default location is the /entry/instrument/NDAttributes group:

```xml
<group name="NDAttributes ndattr_default="true">
  <attribute name="NX_class" source="constant" value="NXcollection" type="string"/>
</group>
```

(continues on next page)
To specify where PVs are written in the HDF5 file, you must create `<dataset />` (or `<attribute />`) elements at the appropriate place in the NeXus HDF5 file layout. See the NeXus manual\(^5\) for placement advice if you are unsure.

You reference each `ndattribute` by its `name` value from the attributes file and use it as the value of the `ndattribute` in the layout file. In this example, `ndattribute="calc1_val"` in the layout file references `name="calc1_val"` in the attributes file and will be identified in the HDF5 file by the name `userCalc1`:

```xml
<dataset
    name="userCalc1"
    source="ndattribute"
    ndattribute="calc1_val"/>
```

Note: A value from the attributes file is only written either in the default location or in the location named by a `<dataset/>` or `<attribute/>` entry in the layout file. Expect problems if you define the same `ndattribute` in more than one place in the layout file.

You can control when a value is written to the file, using `when=""` in the layout file. This can be set to one of these values: `OnFileOpen`, `OnFileClose`

Such as:

```xml
<dataset
    name="userCalc1"
    source="ndattribute"
    ndattribute="calc1_val"
    when="OnFileOpen"/>
```

or:

```xml
<attribute
    name="exposure_s"
    source="ndattribute"
    ndattribute="AcquireTime"
    when="OnFileClose"/>
```

**additional configuration**

Additional configurations of the EPICS Area Detector and the HDF5 File Plugin are done using the EPICS screens (shown here using caQtDM\(^6\)):

Additional configuration on the **ADBase** screen:

- Set **Image mode** to “Single”
- Set **Exposure time** as you wish
- Set **# Images** to 1
- for testing, it is common to bin the data to reduce the image size

---

\(^5\) NeXus manual: https://manual.nexusformat.org/
\(^6\) caQtDM: http://epics.web.psi.ch/software/caqtdm/
Fig. 14: ADBase and NDFileHDF5 configuration screens

- The full path to the attributes.xml file goes in the bottom/left File box

Additional configuration on the NDFileHDF5 screen:
- Set the File path and “File name” to your choice.
- Set Auto save to “Yes”.
- Set Compression to “zlib” if you wish (optional)
- Set Enable to “Enable” or the HDF5 plugin won’t get images to write!
- Set Callbacks block to “Yes” if you want to wait for HDF5 files to finish writing before collecting the next image
- The full path to the layout.xml file goes into the bottom/right XML File name box
- Leave the Attributes file box empty in this screen.

When you enter the names of these files in the configuration screen boxes, AreaDetector will check the files for errors and let you know.

Example view

We collected data for one image, /tmp/mrinal_001.h5, in the HDF5 file provided in the downloads section. You may notice that the values for calc1_val and calc2_val were arrays rather than single values. That was due to an error in the original attributes.xml file, which had type="PARAM" instead of type="EPICS_PV". This has been fixed in the attributes.xml file presented here.
Python code to store an image in a NeXus file

Suppose you want to write area detector images into NeXus HDF5 files python code. Let’s assume you have the image already in memory in a numpy array, perhaps from reading a TIFF file or from an EPICS PV using PyEpics. The file write_nexus_file.py (provided below) reads an image from the sim detector and writes it to a NeXus HDF5 data file, along with some additional metadata.

using the h5py package

This example uses the h5py package to write the HDF file.

```python
import numpy as np
import h5py
import datetime

def write_nexus_file(fname, image, md={}):
    
    # write the image to a NeXus HDF5 data file
    
    Parameters
    ----------
    fname : str
        name of the file (relative or absolute) to be written
    image : numpy array
        the image data
    md : dictionary
        key: value where value is something that can be written by h5py
        (such as str, int, float, numpy array, ...)
    
    nexus = h5py.File(fname, "w")
    nexus.attrs["filename"] = fname
    nexus.attrs["file_time"] = datetime.datetime.now().astimezone().isoformat()
    nexus.attrs["creator"] = "write_nexus_file()"
    nexus.attrs["H5PY_VERSION"] = h5py.__version__

    # /entry
    nxentry = nexus.create_group("entry")
    nxentry.attrs["NX_class"] = "NXentry"
    nexus.attrs["default"] = nxentry.name

    # /entry/instrument
    nxinstrument = nxentry.create_group("instrument")
    nxinstrument.attrs["NX_class"] = "NXinstrument"

    # /entry/instrument/detector
    nxdetector = nxinstrument.create_group("detector")
    nxdetector.attrs["NX_class"] = "NXdetector"

    # /entry/instrument/detector/image
    ds = nxdetector.create_dataset("image", data=image, compression="gzip")
    ds.attrs["units"] = "counts"
```

(continues on next page)
ds.attrs["target"] = "/entry/instrument/detector/image"

# /entry/data
nxdata = nxentry.create_group("data")
nxdata.attrs["NX_class"] = "NXdata"
nxentry.attrs["default"] = nxdata.name

# /entry/data/data --> /entry/instrument/detector/image
nxdata["data"] = nexus["/entry/instrument/detector/image"]
nxdata.attrs["signal"] = "data"

if len(md) > 0:
    # /entry/instrument/metadata (optional, for metadata)
    metadata = nxinstrument.create_group("metadata")
    metadata.attrs["NX_class"] = "NXcollection"
    for k, v in md.items():
        try:
            metadata.create_dataset(k, data=v)
        except Exception:
            metadata.create_dataset(k, data=str(v))

nexus.close()

if __name__ == "__main__":
    """demonstrate how to use this code""
    import epics
    prefix = "13SIM1:"
    img = epics.caget(prefix+"image1:ArrayData")
    size_x = epics.caget(prefix+"cam1:ArraySizeX_RBV")
    size_y = epics.caget(prefix+"cam1:ArraySizeY_RBV")
    # edit the full image for just the binned data
    img = img[:size_x*size_y].reshape((size_x, size_y))
    extra_information = dict(
        unique_id = epics.caget(prefix+"image1:UniqueId_RBV"),
        size_x = size_x,
        size_y = size_y,
        detector_state = epics.caget(prefix+"cam1:DetectorState_RBV"),
        bitcoin_value="15000",
    )
    write_nexus_file("example.h5", img, md=extra_information)

The output from that code is given in the example.h5 file. It has this tree structure:

```
example.h5 : NeXus data file
@H5PY_VERSION = "3.6.0"
@creator = "write_nexus_file()"
@default = "entry"
@file_time = "2022-03-07 14:34:04.418733"
@filename = "example.h5"
entry:NXentry
```

(continues on next page)
```python
@NX_class = "NXentry"
@default = "data"
data:NXdata
    @NX_class = "NXdata"
    @signal = "data"
data --> /entry/instrument/detector/image
instrument:NXinstrument
    @NX_class = "NXinstrument"
detector:NXdetector
    @NX_class = "NXdetector"
image:NX_UINT8[1024,1024] = __array
    __array = [
        [76, 77, 78, '...', 75]
        [77, 78, 79, '...', 76]
        [78, 79, 80, '...', 77]
        ...
        [75, 76, 77, '...', 74]
    ]
    @target = "/entry/instrument/detector/image"
    @units = "counts"
metadata:NXcollection
    @NX_class = "NXcollection"
    bitcoin_value:NX_CHAR = b'15000'
detector_state:NX_INT64[] =
size_x:NX_INT64[] =
size_y:NX_INT64[] =
unique_id:NX_INT64[] =
```

**Note:** Alternatively, the metadata shown in this example might be placed in the `/entry/instrument/detector` (NXdetector) group along with the image data since it provides image-related information such as size.

In the interest of keeping this example simpler and similar to the one above using the HDF5 File Writing Plugin, the metadata has been written into a NXcollection group at `/entry/instrument/metadata` location. (Compare with the NXcollection group `/entry/instrument/NDAttributes` above.)

### using the `nexusformat` package

The `nexusformat` package for Python simplifies the work to create a NeXus file. Rewriting the above code using `nexusformat`:

```python
import numpy as np
from nexusformat.nexus import *

def write_nexus_file(fname, image, md={}):
    ""
    write the image to a NeXus HDF5 data file
    ""
```

---

8 nexusformat: This Python package is described on the NeXPy web site

### 2.3. Examples for Specific Instruments

133
Parameters
----------

fname : str
   name of the file (relative or absolute) to be written
image : numpy array
   the image data
md : dictionary
   key: value where value is something that can be written by h5py
   (such as str, int, float, numpy array, ...)

nx = NXroot()
xn['/entry'] = NXentry(NXinstrument(NXdetector()))
xn['entry/instrument/detector/image'] = NXfield(image, units='counts',
   compression='gzip')
xn['entry/data'] = NXdata()
xn['entry/data'].makelink(nx['entry/instrument/detector/image'])
xn['entry/data'].nxsignal = nx['entry/data/image']

if len(md) > 0:
   # /entry/instrument/metadata (optional, for metadata)
   metadata = nx['/entry/instrument/metadata'] = NXcollection()
   for k, v in md.items():
      metadata[k] = v

nx.save(fname, 'w')

if __name__ == '__main__':
   """demonstrate how to use this code""
   import epics
   prefix = "13SIM1:"
   img = epics.caget(prefix+"image1:ArrayData")
   size_x = epics.caget(prefix+"cam1:ArraySizeX_RBV")
   size_y = epics.caget(prefix+"cam1:ArraySizeY_RBV")
   # edit the full image for just the binned data
   img = img[:size_x*size_y].reshape((size_x, size_y))
   extra_information = dict(
      unique_id = epics.caget(prefix+"image1:UniqueId_RBV"),
      size_x = size_x,
      size_y = size_y,
      detector_state = epics.caget(prefix+"cam1:DetectorState_RBV"),
      bitcoin_value="15000",
   )
   write_nexus_file("example.h5", img, md=extra_information)
Visualization

You can visualize the HDF5 files with several programs, such as: hdfview\textsuperscript{9}, nexpy\textsuperscript{10}, or pymca\textsuperscript{11}. Views of the test image shown using NeXPy (from the HDF5 file) and caQtDM (the image from EPICS) are shown.

Get the installation instructions for any of these programs from a web search. Other data analysis programs such as MatLab, IgorPro, and IDL can also read HDF5 files but you might have to work a bit more to get the data to a plot.

Downloads

<table>
<thead>
<tr>
<th>file</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes.xml</td>
<td>The attributes file</td>
</tr>
<tr>
<td>layout.xml</td>
<td>The layout file</td>
</tr>
<tr>
<td>mrinal_001.h5</td>
<td>example NeXus HDF5 file written from EPICS</td>
</tr>
<tr>
<td>write_nexus_file.py</td>
<td>Python code to get images from EPICS and write a NeXus file</td>
</tr>
<tr>
<td>write_nexus_file2.py</td>
<td>write_nexus_file.py rewritten with nexusformat package</td>
</tr>
<tr>
<td>example.h5</td>
<td>example NeXus HDF5 file written from Python</td>
</tr>
</tbody>
</table>

\textsuperscript{9} hdfview: https://support.hdfgroup.org/products/java/hdfview/

\textsuperscript{10} nexpy: https://nexpy.github.io/nexpy/

\textsuperscript{11} pymca: http://pymca.sourceforge.net/
Footnotes

2.4 Other tools to handle NeXus data files

The number of tools that read NeXus data files, either for general use or to read a specific application definition, is growing. Many of these are open source and so also serve as code examples. In the section NeXus Utilities, we describe many applications and software packages that can read, write, browse, and use NeXus data files. Examples of code (mostly from the NeXus community) that read NeXus data are listed in section Language APIs for NeXus and HDF5.

The NIAC welcomes your continued contributions to this documentation.
3.1 Introduction to NeXus definitions

While the design principles of NeXus are explained in the NeXus: User Manual, this Reference Documentation specifies all allowed base classes and all standardized application definitions. Furthermore, it also contains contributed definitions of new bases classes or application definitions that are currently under review.

Base class definitions and application definitions have basically the same structure, but different semantics:

- Base class definitions define the complete set of terms that might be used in an instance of that class.
- Application definitions define the minimum set of terms that must be used in an instance of that class.

Base classes and application definitions are specified using a domain-specific XML scheme, the NXDL: The NeXus Definition Language.

3.1.1 Overview of NeXus definitions

For each class definition, the documentation is derived from content provided in the NXDL specification.

The documentation for each class consists of sections describing the Status, Description, table of Symbols (if defined), other NeXus base class Groups cited, an annotated Structure, and a link to the NXDL Source (XML) file.

Each of the NXDL files has its own tag in the version repository. Such as NXcrystal-1.0 is tagged in GitHub and accessible via URL: https://github.com/nexusformat/definitions/releases/tag/NXcrystal-1.0
Description

General documentation if this NXDL file.

Symbols table

The Symbols table describes keywords used in this NXDL file to designate array dimensions. For reasons of avoiding naming collisions and to facilitate readability and comprehension for those whom are new to an NXDL file, the following guidelines are strongly encouraged:

- All symbols used in the application definition are defined in a single Symbols table.
- The name of a symbol uses camel case without any white space or underscores.

Examples:

- \textit{nP}: Total number of scan points
- \textit{nE}: Number of photon energies scanned
- \textit{nFrames}: Number of frames
- \textit{detectorRank}: Rank of data array provided by the detector for a single measurement

- the Symbols table appears early in the .nxdl file above the NXentry group

Example from NXtomo.nxdl.xml

```
<definition name="NXtomo" 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"
>
    <symbols>
        <doc>
            These symbols will be used below to coordinate datasets with the same shape.
        </doc>
        <symbol name="nFrames">
            <doc>Number of frames</doc>
        </symbol>
        <symbol name="xSize">
            <doc>Number of pixels in X direction</doc>
        </symbol>
        <symbol name="ySize">
            <doc>Number of pixels in Y direction</doc>
        </symbol>
    </symbols>
    <doc>
        This is the application definition for x-ray or neutron tomography raw data.
        In tomography a number of dark field images are measured, some bright field images and, of course the sample.
        In order to distinguish between them images carry a image_key.
    </doc>
</definition>
```
Annotated Structure

A representation of the basic structure (groups, fields, dimensions, attributes, and links) is prepared for each NXDL specification. Indentation shows nested structure. Attributes are prepended with the @ symbol. Links use the characters -> to represent the path to the intended source of the information.

Indentation is used to indicate nesting of subgroups (a feature common to application definitions). Within each indentation level, NeXus fields are listed first in the order presented in the NXDL file, then groups. Attributes are listed after the documentation of each item and are prefixed with the letter @ (do not use the @ symbol in the actual attribute name). The name of each item is in bold, followed by either optional or required and then the NXDL base class name (for groups) or the NeXus data type (for fields). If units are to be provided with the field, the type of the units is described, such as NX_DATE_TIME.

NeXus Links (these specifications are typically present only in application definitions) are described by a local name, the text ->, then a suggested path to the source item to be linked to the local name.

Names (groups, fields, links, and attributes)

Name of the item. Since name needs to be restricted to valid program variable names, no “-” characters can be allowed. Name must satisfy both HDF and XML naming.

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 the configuration (optional) attribute while the thumbnail element has the mime_type (optional) attribute.

For groups, the name may not be declared in the NXDL specification. In such instances, the value shown in parentheses in the Name and Attributes column is a suggestion, obtained from the group by removing the “NX” prefix. See NXentry for examples.

When the name is allowed to be flexible (the exact name given by this NXDL specification is not required but is set at the time the HDF file is written), the flexible part of the name will be written in all capital letters. For example, in the NXdata group, the DATA, VARIABLE, and VARIABLE_errors fields are flexible.
NeXus data type

Type of data to be represented by this variable. The type is one of those specified in NXDL: The NeXus Definition Language. 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.

For fields, the data type may not be specified in the NXDL file. The default data type is NX_CHAR. See NXdata for examples.

Units

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

Description

A simple text description of the field. No markup or formatting is allowed.

<table>
<thead>
<tr>
<th>NXDL element type</th>
<th>minOccurs</th>
<th>maxOccurs</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>1</td>
<td>unbounded</td>
</tr>
<tr>
<td>field</td>
<td>1</td>
<td>unbounded</td>
</tr>
<tr>
<td>attribute</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Choice

The choice element allows one to create a group with a defined name that is one specific NXDL base class from a defined list of possibilities

In some cases when creating an application definition, more than one choice of base class might be used to define a particular subgroup. For this particular situation, the choice was added to the NeXus NXDL Schema.

In this example fragment of an NXDL application definition, the pixel_shape could be represented by either NXoff_geometry or NXcylindrical_geometry.

```
<choice name="pixel_shape">
  <group type="NXoff_geometry">
    <doc>
      Shape description of each pixel. Use only if all pixels in the detector are of uniform shape.
    </doc>
  </group>
  <group type="NXcylindrical_geometry">
    <doc>
      Shape description of each pixel. Use only if all pixels in the detector are of uniform shape and require being described by cylinders.
    </doc>
  </group>
</choice>
```

1 For NXDL base classes, minOccurs=0 is the default, for NXDL application definitions and contributed definitions, minOccurs=1 is the default. In all cases, the minOccurs attribute in the NXDL file will override the default for that element (group, field, attribute, or link).
The `@name` attribute of the `choice` element specifies the name that will appear in the HDF5 data file using one of the groups listed within the choice. Thus, it is not necessary to specify the name in each group. (At some point, the NXDL Schema may be modified to enforce this rule.)

A `choice` element may be used wherever a `group` element is used. It must have at least two groups listed (otherwise, it would not be useful).

## 3.2 NXDL: The NeXus Definition Language

Information in NeXus data files is arranged by a set of rules. These rules facilitate the exchange of data between scientists and software by standardizing common terms such as the way engineering units are described and the names for common things and the way that arrays are described and stored.

The set of rules for storing information in NeXus data files is declared using the NeXus Definition Language. NXDL itself is governed by a set of rules (a schema) that should simplify learning the few terms in NXDL. In fact, the NXDL rules, written as an XML Schema, are machine-readable using industry-standard and widely-available software tools for XML files such as `xsltproc` and `xmllint`. This chapter describes the rules and terms from which NXDL files are constructed.

### 3.2.1 Introduction

NeXus Definition Language (NXDL) files allow scientists to define the nomenclature and arrangement of information in NeXus data files. These NXDL files can be specific to a scientific discipline such as tomography or small-angle scattering, specific analysis or data reduction software, or even to define another component (base class) used to design and build NeXus data files.

In addition to this chapter and the `Tutorial` chapter, look at the set of NeXus NXDL files to learn how to read and write NXDL files. These files are available from the NeXus definitions repository and are most easily viewed on GitHub: https://github.com/nexusformat/definitions in the `base_classes`, `applications`, and `contributed` directories. The rules (expressed as XML Schema) for NXDL files may also be viewed from this URL. See the files `nxdl.xsd` for the main XML Schema and `nxdlTypes.xsd` for the listings of allowed data types and categories of units allowed in NXDL files.

NXDL files can be checked (validated) for syntax and content. With validation, scientists can be certain their definitions will be free of syntax errors. Since NXDL is based on the XML standard, there are many editing programs\(^1\) available to ensure that the files are well-formed.\(^2\) There are many standard tools such as `xmllint` and `xsltproc` that can process XML files. Further, NXDL files are backed by a set of rules (an XML Schema) that define the language and can be used to check that an NXDL file is both correct by syntax and valid by the NeXus rules.

NXDL files are machine-readable. This enables their automated conversion into schema files that can be used, in combination with other NXDL files, to validate NeXus data files. In fact, all of the tables in the *Class Definitions* Chapter have been generated directly from the NXDL files.

---

**Tip:** Use the reST anchors when writing documentation in NXDL source files. Since the anchors have no title or caption associated, you will need to supply text with the reference, such as:

```plaintext
::ref::this text will appear <anchor>`
```
The language of NXDL files is intentionally quite small, to provide only that which is necessary to describe scientific data structures (or to establish the necessary XML structures). Rather than have scientists prepare XML Schema files directly, NXDL was designed to reduce the jargon necessary to define the structure of data files. The two principle objects in NXDL files are: group and field. Documentation (doc) is optional for any NXDL component. Either of these objects may have additional attributes that contribute simple metadata.

The Class Definitions Chapter lists the various classes from which a NeXus file is constructed. These classes 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, refer to the class definitions for the standard nomenclature. 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.

**NXDL Elements and Field Types**

The documentation in this section has been obtained directly from the NXDL Schema file: nxdl.xsd. First, the basic elements are defined in alphabetical order. Attributes to an element are indicated immediately following the element and are preceded with an “@” symbol, such as @attribute. Then, the common data types used within the NXDL specification are defined. Pay particular attention to the rules for validItemName and validNXClassName.

**NXDL Elements**

**attribute**

An attribute element can only be a child of a field or group element. It is used to define attribute elements to be used and their data types and possibly an enumeration of allowed values.

For more details, see: attributeType

**choice**

A choice element is used when a named group might take one of several possible NeXus base classes. Logically, it must have at least two group children.

For more details, see: choiceType

**definition**

A definition element can only be used at the root level of an NXDL specification. Note: Due to the large number of attributes of the definition element, they have been omitted from the figure below.

For more details, see: definition, definitionType, and definitionTypeAttr
The dimensions element describes the shape of an array. It is used only as a child of a field element. For more details, see: dimensionsType

doc

A doc element can be a child of most NXDL elements. In most cases, the content of the doc element will also become part of the NeXus manual.

element
  {any}:

In documentation, it may be useful to use an element that is not directly specified by the NXDL language. The any element here says that one can use any element at all in a doc element and NXDL will not process it but pass it through. For more details, see: docType
Fig. 2: Graphical representation of the NXDL definition element
3.2. NXDL: The NeXus Definition Language

Fig. 3: Graphical representation of the NXDL dimensions element

Fig. 4: Graphical representation of the NXDL doc element
**enumeration**

An enumeration element can only be a child of a field or attribute element. It is used to restrict the available choices to a predefined list, such as to control varieties in spelling of a controversial word (such as *metre* vs. *meter*).

For more details, see: *enumerationType*

![Fig. 5: Graphical representation of the NXDL enumeration element](image)

**field**

The field element provides the value of a named item. Many different attributes are available to further define the field. Some of the attributes are not allowed to be used together (such as axes and axis); see the documentation of each for details. It is used only as a child of a group element.

For more details, see: *fieldType*

**group**

A group element can only be a child of a definition or group element. It describes a common level of organization in a NeXus data file, similar to a subdirectory in a file directory tree.

For more details, see: *groupType*

**link**

A link element can only be a child of a definition, field, or group element. It describes the path to the original source of the parent definition, field, or group.

For more details, see: *linkType*

**symbols**

A symbols element can only be a child of a definition element. It defines the array index symbols to be used when defining arrays as field elements with common dimensions and lengths.

For more details, see: *symbolsType*
3.2. NXDL: The NeXus Definition Language

Fig. 6: Graphical representation of the NXDL field element
Fig. 7: Graphical representation of the NXDL group element
Fig. 8: Graphical representation of the NXDL link element

Fig. 9: Graphical representation of the NXDL symbols element
NXDL Field Types (internal)

Field types that define the NXDL language are described here. These data types are defined in the XSD Schema (nxdl.xsd) and are used in various parts of the Schema to define common structures or to simplify a complicated entry. While the data types are not intended for use in NXDL specifications, they define structures that may be used in NXDL specifications.

attributeType

Any new group or field may expect or require some common attributes.
(This data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)

Attributes of attributeType

@name

Name of the attribute (unique within the enclosing group).

@optional

Is this attribute optional (if true) or required (if false)?

@recommended

A synonym for optional, but with the recommendation that this attribute be specified.

@type

Type of the attribute. For group specifications, the class name. For field or attribute specifications, the NXDL field type.

Elements of attributeType

dimensions

dimensions of an attribute with data value(s) in a NeXus file
doc

Description of this attribute. This documentation will go into the manual.

enumeration

An enumeration specifies the values to be used.

definition

A definition element is the group at the root of every NXDL specification. It may only appear at the root of an NXDL file and must only appear once for the NXDL to be well-formed.

definitionType

A definition is the root element of every NXDL definition. It may only appear at the root of an NXDL file and must only appear once for the NXDL to be well-formed.

The definitionType defines the documentation, attributes, fields, and groups that will be used as children of the definition element. Could contain these elements:

• attribute
• doc
• field
• group
• link

Note that a definition element also includes the definitions of the basicComponent data type. (The definitionType data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)

Note that the first line of text in a doc element in a definition is used as a summary in the manual. Follow the pattern as shown in the base class NXDL files.

Attributes of definitionType

@category

NXDL base definitions define the dictionary of terms to use for these components. All terms in a base definition are optional. NXDL application definitions define what is required for a scientific interest. All terms in an application definition are required. NXDL contributed definitions may be considered either base or applications. Contributed definitions must indicate their intended use, either as a base class or as an application definition.
@extends

The extends attribute allows this definition to subclass from another NXDL, otherwise extends="NXobject" should be used.

@ignoreExtraAttributes

Only validate known attributes; do not not warn about unknowns. The ignoreExtraAttributes attribute is a flag to the process of validating NeXus data files. By setting ignoreExtraAttributes="true", presence of any undefined attributes in this class will not generate warnings during validation. Normally, validation will check all the attributes against their definition in the NeXus base classes and application definitions. Any items found that do not match the definition in the NXDL will generate a warning message.

The ignoreExtraAttributes attribute should be used sparingly!

@ignoreExtraFields

Only validate known fields; do not not warn about unknowns. The ignoreExtraFields attribute is a flag to the process of validating NeXus data files. By setting ignoreExtraFields="true", presence of any undefined fields in this class will not generate warnings during validation. Normally, validation will check all the fields against their definition in the NeXus base classes and application definitions. Any items found that do not match the definition in the NXDL will generate a warning message.

The ignoreExtraFields attribute should be used sparingly!

@ignoreExtraGroups

Only validate known groups; do not not warn about unknowns. The ignoreExtraGroups attribute is a flag to the process of validating NeXus data files. By setting ignoreExtraGroups="true", presence of any undefined groups in this class will not generate warnings during validation. Normally, validation will check all the groups against their definition in the NeXus base classes and application definitions. Any items found that do not match the definition in the NXDL will generate a warning message.

The ignoreExtraGroups attribute should be used sparingly!

@name

The name of this NXDL file (case sensitive without the file extension). The name must be unique amongst all the NeXus base class, application, and contributed definitions. For the class to be adopted by the NIAC, the first two letters must be "NX" (in uppercase). Any other use must not begin with "NX" in any combination of upper or lower case.
@restricts

The restricts attribute is a flag to the data validation. When restricts="1", any non-standard component found (and checked for validity against this NXDL specification) in a NeXus data file will be flagged as an error. If the restricts attribute is not present, any such situations will produce a warning.

@svnid

(2014-08-19: deprecated since switch to GitHub version control) The identifier string from the subversion revision control system. This reports the time stamp and the revision number of this file.

@type

Must be type="group"

Elements of definitionType

symbols

Use a symbols list to define each of the mnemonics that represent the length of each dimension in a vector or array.

Groups under definitionType

In addition to an optional symbols list, a definition may contain any of the items allowed in a group.

definitionTypeAttr

Prescribes the allowed values for definition type attribute. (This data type is used internally in the NXDL schema to define a data type.)

The value may be any one from this list only:

- group
- definition

dimensionsType

dimensions of a data element in a NeXus file (This data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)
Attributes of dimensionsType

@rank

Rank (number of dimensions) of the data structure.
Value could be either an unsigned integer or a symbol as defined in the symbol table of the NXDL file.
For example: a[5] has rank="1" while b[8, 5, 6, 4] has rank="4". See https://en.wikipedia.org/wiki/Rank_%28computer_programming%29 for more details.

Elements of dimensionsType

dim

Specify the parameters for each index of the dimensions element with a dim element. The number of dim entries should be equal to the rank of the array. For example, these terms describe a 2-D array with lengths (nsurf, nwl):

```

```

The value attribute is used by NXDL and also by the NeXus data file validation tools to associate and coordinate the same array length across multiple fields in a group.

@incr

Deprecated: 2016-11-23 telco (https://github.com/nexusformat/definitions/issues/330)
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.

@index

Number or symbol indicating which axis (subscript) is being described, ranging from 1 up to rank (rank of the data structure). For example, given an array A[i,j,k], index="1" would refer to the i axis (subscript). (NXdata uses index="0" to indicate a situation when the specific index is not known a priori.)

@ref

Deprecated: 2016-11-23 telco (https://github.com/nexusformat/definitions/issues/330)
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.
@refindex

Deprecated: 2016-11-23 telco (https://github.com/nexusformat/definitions/issues/330)

The dimension specification is the same as the refindex axis within the ref field. Requires ref attribute to be present.

@required

This dimension is required (true: default) or not required (false).

The default value is true.

When required="false" is specified, all subsequent <dim nodes (with higher index value) must also have required="false".

@value

Integer length (number of values), or mnemonic symbol representing the length of this axis.

doc

Documentation might be necessary to describe how the parts of the dimensions element are to be used.

docType

NXDL allows for documentation on most elements using the doc element. The documentation is useful in several contexts. The documentation will be rendered in the manual. Documentation is provided as tooltips by some XML editors when editing NXDL files. Simple documentation can be typed directly in the NXDL:

<table>
<thead>
<tr>
<th>Descriptive name of sample</th>
</tr>
</thead>
</table>

This is suitable for basic descriptions that do not need extra formatting such as a bullet-list or a table. For more advanced control, use the rules of restructured text, such as in the NXdetector specification. Refer to examples in the NeXus base class NXDL files such as NXdata.

Could contain these elements:

- *any*

(This data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)

Note: For documentation of definition elements, the first line of text in a doc is used as a summary in the manual. Follow the pattern as shown in the base class NXDL files.)
enumerationType

An enumeration restricts the values allowed for a specification. Each value is specified using an item element, such as: `<item value="Synchrotron X-ray Source" />`. Could contain these elements:

- doc
- item

(This data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)

<table>
<thead>
<tr>
<th>source operating mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>for storage rings</td>
</tr>
<tr>
<td>for storage rings</td>
</tr>
</tbody>
</table>

Elements of enumerationType

item

One of the prescribed values. Use the value attribute.

Defines the value of one selection for an enumeration list. Each enumerated item must have a value (it cannot have an empty text node).

@value

The value of value of an enumItem is defined as an attribute rather than a name.

doc

Individual items can be documented but this documentation might not be printed in the NeXus Reference Guide.

fieldType

A field declares a new element in the component being defined. A field is synonymous with the HDF4 SDS (Scientific Data Set) and the HDF5 dataset terms. Could contain these elements:

- attribute
- dimensions
- doc
- enumeration

Note that a field element also includes the definitions of the basicComponent data type. (The fieldType data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)
@axes

NOTE: Use of the axes attribute for a field is discouraged. It is for legacy support. You should use the axes group attribute (such as in NXdata) instead.

This attribute contains a string array that defines the independent data fields used in the default plot for all of the dimensions of the signal field (the signal field is the field in this group that is named by the signal attribute of this group).

When there is only one item in the string array, it is acceptable to set the value to the one string. In such case, it is not necessary to make it an array of one string.

Presence of the axes attribute means this field is an ordinate.

@axis

NOTE: Use of this attribute is discouraged. It is for legacy support. You should use the axes group attribute (such as in NXdata) instead.

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

The attribute value is an integer indicating 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 (abscissae) 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.

@data_offset

The stride and data_offset attributes are used together to index the array of data items in a multi-dimensional array. They may be used as an alternative method to address a data array that is not stored in the standard NeXus method of “C” order.

The data_offset attribute determines the starting coordinates of the data array for each dimension.


The data_offset attribute contains a comma-separated list of integers. (In addition to the required comma delimiter, whitespace is also allowed to improve readability.) The number of items in the list is equal to the rank of the data being stored. The value of each item is the offset in the array of the first data item of that subscript of the array.
@interpretation

This instructs the consumer of the data what the last dimensions of the data are. It allows plotting software to work out the natural way of displaying the data.

For example a single-element, energy-resolving, fluorescence detector with 512 bins should have interpretation="spectrum". If the detector is scanned over a 512 x 512 spatial grid, the data reported will be of dimensions: 512 x 512 x 512. In this example, the initial plotting representation should default to data of the same dimensions of a 512 x 512 pixel image detector where the images where taken at 512 different pressure values.

In simple terms, the allowed values mean:

- scalar = 0-D data to be plotted
- scaler = DEPRECATED, use scalar
- spectrum = 1-D data to be plotted
- image = 2-D data to be plotted
- rgb-image = 3-D data to be plotted
- rgba-image = 3-D data to be plotted
- hsl-image = 3-D data to be plotted
- hsla-image = 3-D data to be plotted
- cmyk-image = 3-D data to be plotted
- vertex = 3-D data to be plotted

@long_name

Descriptive name for this field (may include whitespace and engineering units). Often, the long_name (when defined) will be used as the axis label on a plot.

@maxOccurs

Defines the maximum number of times this element may be used. Its value is confined to zero or greater. Must be greater than or equal to the value for the “minOccurs” attribute. A value of “unbounded” is allowed.

@minOccurs

Defines the minimum number of times this field may be used. Its value is confined to zero or greater. Must be less than or equal to the value for the “maxOccurs” attribute.
@nameType

This interprets the name attribute as: * specified = use as specified * any = can be any name not already used in group

@optional

A synonym for minOccurs=0.

@primary

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 abscissa.

@recommended

A synonym for optional, but with the recommendation that this field be specified.

@signal

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.

@stride

The stride and data_offset attributes are used together to index the array of data items in a multidimensional array. They may be used as an alternative method to address a data array that is not stored in the standard NeXus method of “C” order.

The stride list chooses array locations from the data array with each value in the stride list determining how many elements to move in each dimension. Setting a value in the stride array to 1 moves to each element in that dimension of the data array, while setting a value of 2 in a location in the stride array moves to every other element in that dimension of the data array. A value in the stride list may be positive to move forward or negative to step backward. A value of zero will not step (and is of no particular use).


The stride attribute contains a comma-separated list of integers. (In addition to the required comma delimiter, whitespace is also allowed to improve readability.) The number of items in the list is equal to the rank of the data being stored. The value of each item is the spacing of the data items in that subscript of the array.
@type

Defines the type of the element as allowed by NeXus.

See here and elsewhere for the complete list of allowed types.

@units

String describing the engineering units. The string should be appropriate for the value and should conform to the NeXus rules for units. Conformance is not validated at this time.

attribute

attributes to be used with this field

dimensions

dimensions of a data element in a NeXus file

enumeration

A field can specify which values are to be used

choiceType

A choice element is used when a named group might take one of several possible NeXus base classes. Logically, it must have at least two group children.

Attributes of choiceType

@name

The name to be applied to the selected child group. None of the child groups should define a @name attribute.

Elements of choiceType

group

NeXus base class that could be used here. The group will take the @name attribute defined by the parent choice element so do not specify the @name attribute of the group here.
**groupType**

A group element refers to the definition of an existing NX object or a locally-defined component. Could contain these elements:

- attribute
- doc
- field
- group
- link

Note that a group element also includes the definitions of the basicComponent data type. (The groupType data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)

**Attributes of groupType**

[@maxOccurs]

Maximum number of times this group is allowed to be present within its parent group. Note each group must have a name attribute that is unique among all group and field declarations within a common parent group.

[@minOccurs]

Minimum number of times this group is allowed to be present within its parent group. Note each group must have a name attribute that is unique among all group and field declarations within a common parent group.

[@name]

A particular scientific application may expect a name of a group element. It is helpful but not required to specify the name attribute in the NXDL file. It is suggested to always specify a name to avoid ambiguity. It is also suggested to derive the name from the type, using an additional number suffix as necessary. For example, consider a data file with only one NXentry. The suggested default name would be entry. For a data file with two or more NXentry groups, the suggested names would be entry1, entry2, ... Alternatively, a scientific application such as small-angle scattering might require a different naming procedure; two different NXaperture groups might be given the names beam_defining_slit and scatter_slit.

[@optional]

A synonym for minOccurs=0.
@recommended

A synonym for optional, but with the recommendation that this group be specified.

@type

The type attribute must contain the name of a NeXus base class, application definition, or contributed definition.

linkType

A link to another item. Use a link to avoid needless repetition of information. (This data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)

@napimount

Group attribute that provides a URL to a group in another file. More information is described in the NeXus Programmers Reference.


@target

Declares the absolute HDF5 address of an existing field or group.

The target attribute is added for NeXus to distinguish the HDF5 path to the original dataset.

Could contain these elements:

- doc

Matching regular expression:

```
(/[a-zA-Z_][\w_]*(:[a-zA-Z_][\w_]*))?+
```

For example, given a /entry/instrument/detector/polar_angle field, link it into the NXdata group (at /entry/data/polar_angle). This would be the NeXus data file structure:

```xml
/: NeXus/HDF5 data file
 /entry: NXentry
  /data: NXdata
   /polar_angle: NX_NUMBER
   @target="/entry/instrument/detector/polar_angle"
  ...
 /instrument: NXinstrument
  /detector: NXdetector
   /polar_angle: NX_NUMBER
   @target="/entry/instrument/detector/
polar_angle"
```
symbolsType

Each symbol has a name and optional documentation. Please provide documentation that indicates what each symbol represents. For example:

<table>
<thead>
<tr>
<th>number of reflecting surfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of wavelengths</td>
</tr>
</tbody>
</table>

Elements of symbolsType

doc

Describe the purpose of this list of symbols. This documentation will go into the manual.

symbol

When multiple field elements share the same dimensions, such as the dimension scales associated with plottable data in an NXdata group, the length of each dimension written in a NeXus data file should be something that can be tested by the data file validation process.

@name

Mnemonic variable name for this array index symbol.

doc

Describe the purpose of the parent symbol. This documentation will go into the manual.

basicComponent

A basicComponent defines the allowed name format and attributes common to all field and group specifications. (This data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)

Attributes of basicComponent

@name

The name attribute is the identifier string for this entity. It is required that name must be unique within the enclosing group. The name must match the regular expression defined by validItemName. (Historical note: Originally, the rule (validItemName) was defined to allow only names that can be represented as valid variable names in most computer languages.)
Elements of basicComponent

doc

Describe this basicComponent and its use. This documentation will go into the manual.

validItemName

Used for allowed names of elements and attributes. Note: No - characters (among others) are allowed and you cannot start or end with a period (.). HDF4 had a 64 character limit on names (possibly including NULL) and the NAPI enforces this via the NX_MAXNAMELEN variable with a 64 character limit (which may be 63 on a practical basis if one considers a NULL terminating byte). (This data type is used internally in the NXDL schema to define a data type.)

NOTE: In some languages, it may be necessary to add a ^ at the start and a $ at the end of the regular expression to constrain the match to an entire line.

The value may be any xs:token that also matches the regular expression:

```
[a-zA-Z0-9-\_][a-zA-Z0-9-\_.]*[a-zA-Z0-9-\_]
```

validNXClassName

Used for allowed names of NX class types (e.g. NXdetector). Note this is not the instance name (e.g. bank1) which is covered by validItemName. (This data type is used internally in the NXDL schema to define a data type.)

The value may be any nx:validItemName that also matches the regular expression:

```
NX.+
```

validTargetName

This is a valid link target - currently it must be an absolute path made up of valid names with the / character delimiter. But we may want to consider allowing ".." (parent of directory) at some point. If the name attribute is helpful, then use it in the path with the syntax of name:type as in these examples:

```
/NXentry/NXinstrument/analyzer:NXcrystal/ef
/NXentry/NXinstrument/monochromator:NXcrystal/ei
/NX_other
```

Must also consider use of name attribute in resolving link targets. (This data type is used internally in the NXDL schema to define a data type.)

From the HDF5 documentation:

> 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 [name:]type, the matching regular expression syntax is written: /

```
[a-zA-Z_][\w_]*([a-zA-Z_][\w_]+)?
```

Note that HDF5 also permits relative path names, such as: GroupA/GroupB/Dataset1 but this is not permitted in the matching regular expression and not supported in NAPI.

The value may be any xs:token that also matches the regular expression:
nonNegativeUnbounded

A `nonNegativeUnbounded` allows values including all positive integers, zero, and the string `unbounded`. (This data type is used internally in the NXDL schema to define a data type.)

**The `xs:string` data type**

The `xs:string` data type can contain characters, line feeds, carriage returns, and tab characters. See [https://www.w3schools.com/xml/schema_dtypes_string.asp](https://www.w3schools.com/xml/schema_dtypes_string.asp) for more details.

**The `xs:token` data type**

The `xs:token` data type is derived from the `xs:string` data type.

The `xs:token` data type also contains characters, but the XML processor will remove line feeds, carriage returns, tabs, leading and trailing spaces, and multiple spaces. See [https://www.w3schools.com/xml/schema_dtypes_string.asp](https://www.w3schools.com/xml/schema_dtypes_string.asp) for more details.

### NXDL Data Types and Units

#### Data Types allowed in NXDL specifications

Data types for use in NXDL describe the expected type of data for a NeXus field or attribute. 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. The default type `NX_CHAR` is applied in cases where a field or attribute is defined in an NXDL specification without explicit assignment of a type.

**ISO8601**

ISO8601 date/time stamp. It is recommended to add an explicit time zone, otherwise the local time zone is assumed per ISO8601. The norm is that if there is no time zone, it is assumed local time, however, when a file moves from one country to another it is undefined. If the local time zone is written, the ambiguity is gone.

**NX_BINARY**

any representation of binary data - if text, line terminator is [CR][LF]

**NX_BOOLEAN**

true/false value ( true | 1 | false | 0 )

**NX_CCOMPLEX**

Compound type cartesian representation of complex numbers (real and imaginary parts) in NeXus.

**NX_CHAR**

The preferred string representation is UTF-8. Both fixed-length strings and variable-length strings are valid. String arrays cannot be used where only a string is expected (title, start_time, end_time, `NX_class` attribute,...). Fields or attributes requiring the use of string arrays will be clearly marked as such (like the `NXdata` attribute `auxiliary_signals`). This is the default field type.

**NX_COMPLEX**

Compound type representation of complex numbers (either cartesian or polar form) in NeXus.

**NX_DATE_TIME**

Alias for the `ISO8601` date/time stamp. It is recommended to add an explicit time zone, otherwise the local time zone is assumed per ISO8601.
NX_FLOAT
any representation of a floating point number

NX_INT
any representation of an integer number

NX_NUMBER
any valid NeXus number representation

NX_PCOMPLEX
Compound type polar representation of complex numbers (amplitude and phase in radians) in NeXus.

NX_POSINT
any representation of a positive integer number (greater than zero)

NX_QUATERNION
Compound type representation of quaternion numbers (real,i,j,k) in NeXus.

NX_UINT
any representation of an unsigned integer number (includes zero)

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 NeXus units section. The values for unit categories are restricted (by an enumeration) to the following table.

NX_ANGLE
units of angle,
example(s): rad

NX_ANY
units for things like logs that aren’t picky on units

NX_AREA
units of area,
example(s): m^2 | barns

NX_CHARGE
units of electrical charge,
example(s): C

NX_COUNT
units of quantity of item(s) such as number of photons, neutrons, pulses, or other counting events

NX_CROSS_SECTION
units of area (alias of NX_AREA),
example(s): barn

NX_CURRENT
units of electrical current,
example(s): A

NX_DIMENSIONLESS
units for fields where the units cancel out (NOTE: not the same as NX_UNITLESS),
example(s): m/m
NX_EMITTANCE
  units of emittance \((\text{length} \times \text{angle})\) of a radiation source,
  example(s): \(\text{nm} \times \text{rad}\)

NX_ENERGY
  units of energy,
  example(s): \(\text{J} \mid \text{keV}\)

NX_FLUX
  units of flux,
  example(s): \(\text{1/s/cm}^2\)

NX_FREQUENCY
  units of frequency,
  example(s): \(\text{Hz}\)

NX_LENGTH
  units of length,
  example(s): \(\text{m}\)

NX_MASS
  units of mass,
  example(s): \(\text{g}\)

NX_MASS_DENSITY
  units of mass density,
  example(s): \(\text{g/cm}^3\)

NX_MOLECULAR_WEIGHT
  units of molecular weight,
  example(s): \(\text{g/mol}\)

NX_PERIOD
  units of time, period of pulsed source (alias to \(\text{NX_TIME}\)),
  example(s): \(\text{us}\)

NX_PER_AREA
  units of \(\text{1/length}^2\),
  example(s): \(\text{1/m}^2\)

NX_PER_LENGTH
  units of \(\text{1/length}\),
  example(s): \(\text{1/m}\)

NX_POWER
  units of power,
  example(s): \(\text{W}\)

NX_PRESSURE
  units of pressure,
  example(s): \(\text{Pa}\)
NX_PULSES
DEPRECATED: see NX_COUNT
units of clock pulses (alias to NX_NUMBER)

NX_SCATTERING_LENGTH_DENSITY
units of scattering length density,
example(s): m/m^3

NX_SOLID_ANGLE
units of solid angle,
example(s): sr | steradian

NX_TEMPERATURE
units of temperature,
example(s): K

NX_TIME
units of time,
example(s): s

NX_TIME_OF_FLIGHT
units of (neutron) time of flight (alias to NX_TIME),
example(s): s

NX_TRANSFORMATION
units of the specified transformation
could be any of these: NX_LENGTH, NX_ANGLE, or NX_UNITLESS
There will be one or more transformations defined by one or more fields for each transformation. The units type NX_TRANSFORMATION designates the particular axis generating a transformation (e.g. a rotation axis or a translation axis or a general axis). NX_TRANSFORMATION designates the units will be appropriate to the type of transformation, indicated in the NXtransformations base class by the transformation_type value:
• NX_LENGTH for translation
• NX_ANGLE for rotation
• NX_UNITLESS for axes for which no transformation type is specified.

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),
example(s): ""

NX_VOLTAGE
units of voltage,
example(s): V

NX_VOLUME
units of volume,
example(s): m^3

NX_WAVELENGTH
units of wavelength,
example(s): angstrom
NX_WAVENUMBER

units of wavenumber or \( Q \),
example(s): \( 1/\text{nm} \) | \( 1/\text{angstrom} \)

NXDL File Organisation

NXDL File Name

In order for the XML machinery to find and link the code in the various files, the name of the file must be composed of the definition name (matching both the spelling and the case) and a “.nxdl.xml” extension. For example, the base class `NXarbitrary_example` should be defined by NXDL code within the `NXarbitrary_example.nxdl.xml` file. Note also that the definition name is stated twice in application definitions, once in the `definition` tag, and again as the value of an `item` contained within the `field` tag that is named “definition”.

Listing 1: `NXarbitrary_example.nxdl.xml`

```xml
<definition name="NXarbitrary_example">
    <!-- later -->
    <field name="definition">
        <doc>Official NeXus NXDL schema to which this file conforms.</doc>
        <enumeration>
            <item value="NXarbitrary_example"/>
        </enumeration>
    </field>
</definition>
```

Documentation Images

Including images (or other related content) in the documentation of NXDL definitions can be very effective for communicating how different parts of the definition interact. To be properly included in the compilation of the NeXus documentation, the extra files must go into a directory having the same name as the definition without the NX prefix. For example, if the `NXarbitrary_example` base class has a `pretty_picture.jpg` image included in its documentation, then the image file should be located by the path (relative to `NXarbitrary_example.xml`) `arbitrary_example/pretty_picture.jpg`.

![Diagram of file structure](image-url)
3.3 NeXus Class Definitions

Definitions of NeXus classes. These are split into base classes (low level objects), application definitions (groupings of objects for a particular technique) and contributed definitions (proposed definitions from the community).

The complete vocabulary of terms used in NeXus NXDL files (names of groups, fields, attributes, and links) is available for download.

Base classes

NeXus base class definitions define the set of terms that might be used in an instance of that class. Consider the base classes as a set of components that are used to construct a data file.

Base class definitions are permissive rather than restrictive. While the terms defined aim to cover most possible use cases, and to codify the spelling and meaning of such terms, the class specifications cannot list all acceptable groups and fields. To be able to progress the NeXus standard, additional data (groups, fields, attributes) are acceptable in NeXus HDF5 data files.

Users are encouraged to find the best defined location in which to place their information. It is understood there is not a predefined place for all possible data.

Validation procedures should treat such additional items (not covered by a base class specification) as notes or warnings rather than errors.

Application Definitions

NeXus application definitions define the minimum set of terms that must be used in an instance of that class. Application definitions also may define terms that are optional in the NeXus data file.

As in base classes (see above), additional terms that are not described by the application definition may be added to data files that incorporate or adhere to application definitions.

Use NeXus links liberally in data files to reduce duplication of data. In application definitions involving raw data, write the raw data in the NXinstrument tree and then link to it from the location(s) defined in the relevant application definition. See figure NeXus Multi Method Hierarchy for an example.

To write a data file with an application definition, start with either a NXentry (or NXsubentry) group\(^1\) and write the name of the application definition in the definition field. Then write data into this group according to the specifications of the application definition.

\(^1\) For data files involving just an application definition, use the NXentry group. Such as this structure:

```xml
<entry>
  <NXentry
   definition="NXsas"
  />
</entry>
```

For files that describe multi-modal data and require use of two or more application definitions (such as NXsas and NXcanSAS), you must place each application definition in a NXsubentry of the NXentry group. Such as this structure:

```xml
<entry>
  <raw>
    <NXsubentry
      definition="NXsas"
    />
  </raw>
  <reduced>
    <NXsubentry
      definition="NXcanSAS"
    />
  </reduced>
  <fluomod>
    <NXsubentry
      definition="NXfluomod"
    />
  </fluomod>
</entry>
```

If you anticipate your data file will eventually require an additional application definition, you should start with each application definition in a NXsubentry group.
Contributed Definitions

NXDL files in the NeXus contributed definitions include propositions from the community for NeXus base classes or application definitions, as well as other NXDL files for long-term archival by NeXus. Consider the contributed definitions as either in incubation or a special case not for general use.

3.3.1 Base Class Definitions

A description of each NeXus base class definition is given. NeXus base class definitions define the set of terms that might be used in an instance of that class. Consider the base classes as a set of components that are used to construct a data file.

**NXaperture**
A beamline aperture. This group is deprecated, use NXslit instead.

**NXattenuator**
A device that reduces the intensity of a beam by attenuation.

**NXbeam**
Properties of the neutron or X-ray beam at a given location.

**NXbeam_stop**
A device that blocks the beam completely, usually to protect a detector.

**NXbending_magnet**
A bending magnet

**NXcapillary**
A capillary lens to focus the X-ray beam.

**NXcite**
A literature reference

**NXcollection**
An unvalidated set of terms, such as the description of a beam line.

**NXcollimator**
A beamline collimator.

**NXcrystal**
A crystal monochromator or analyzer.

**NXcylindrical_geometry**
Geometry description for cylindrical shapes.

**NXdata**
**NXdata** describes the plottable data and related dimension scales.

**NXdetector**
A detector, detector bank, or multidetector.

**NXdetector_group**
Logical grouping of detectors. When used, describes a group of detectors.

**NXdetector_module**
Geometry and logical description of a detector module. When used, child group to NXdetector.

**NXdisk_chopper**
A device blocking the beam in a temporal periodic pattern.

**NXentry**
(required) **NXentry** describes the measurement.
**NXenvironment**
Parameters for controlling external conditions

**NXevent_data**
NXevent_data is a special group for storing data from neutron

**NXfermi_chopper**
A Fermi chopper, possibly with curved slits.

**NXfilter**
For band pass beam filters.

**NXflipper**
A spin flipper.

**NXfresnel_zone_plate**
A fresnel zone plate

**NXgeometry**
legacy class - recommend to use **NXtransformations** now

**NXgrating**
A diffraction grating, as could be used in a soft X-ray monochromator

**NXguide**
A neutron optical element to direct the path of the beam.

**NXinsertion_device**
An insertion device, as used in a synchrotron light source.

**NXinstrument**
Collection of the components of the instrument or beamline.

**NXlog**
Information recorded as a function of time.

**NXmirror**
A beamline mirror or supermirror.

**NXmoderator**
A neutron moderator

**NXmonitor**
A monitor of incident beam data.

**NXmonochromator**
A wavelength defining device.

**NXnote**
Any additional freeform information not covered by the other base classes.

**NXobject**
This is the base object of NeXus

**NXoff_geometry**
Geometry (shape) description.

**NXorientation**
legacy class - recommend to use **NXtransformations** now

**NXparameters**
Container for parameters, usually used in processing or analysis.
**NXpdb**
A NeXus transliteration of a PDB file, to be validated only as a PDB

**NXpinhole**
A simple pinhole.

**NXpolarizer**
A spin polarizer.

**NXpositioner**
A generic positioner such as a motor or piezo-electric transducer.

**NXprocess**
Document an event of data processing, reconstruction, or analysis for this data.

**NXreflections**
Reflection data from diffraction experiments

**NXroot**
Definition of the root NeXus group.

**NXsample**
Any information on the sample.

**NXsample_component**
One group like this per component can be recorded For a sample consisting of multiple components.

**NXsensor**
A sensor used to monitor an external condition

**NXshape**
legacy class - (used by NXgeometry) - the shape and size of a component.

**NXslit**
A simple slit.

**NXsource**
The neutron or x-ray storage ring/facility.

**NXsubentry**
Group of multiple application definitions for “multi-modal” (e.g. SAXS/WAXS) measurements.

**NXtransformations**
Collection of axis-based translations and rotations to describe a geometry.

**NXtranslation**
legacy class - (used by NXgeometry) - general spatial location of a component.

**NXuser**
Contact information for a user.

**NXvelocity_selector**
A neutron velocity selector

**NXxraylens**
An X-ray lens, typically at a synchrotron X-ray beam line.
NXaperture

Status:

base class, extends NXobject

Description:

A beamline aperture. This group is deprecated, use NXslit instead.

Symbols:

No symbol table

Groups cited:

NXgeometry, NXnote, NXtransformations

Structure:

@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

depends_on: (optional) NX_CHAR

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

The reference point of the aperture is its center in the x and y axis. The reference point on the z axis is the surface of the aperture pointing towards the source.

In complex (asymmetric) geometries an NXoff_geometry group can be used to provide an unambiguous reference.

material: (optional) NX_CHAR

Absorbing material of the aperture

description: (optional) NX_CHAR

Description of aperture

TRANSFORMATIONS: (optional) NXtransformations

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

GEOMETRY: (optional) NXgeometry
DEPRECATED: Use the field `depends_on` and `NXtransformations` to position the aperture and `NXoff_geometry` to describe its shape.

**BLADE_GEOMETRY**: (optional) `NXgeometry`

DEPRECATED: Use `NXoff_geometry` instead to describe the shape of the aperture.

**NOTE**: (optional) `NXnote`

describe any additional information in a note*

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXaperture/BLADE_GEOMETRY-group`
- `/NXaperture/depends_on-field`
- `/NXaperture/description-field`
- `/NXaperture/GEOMETRY-group`
- `/NXaperture/material-field`
- `/NXaperture/NODE-group`
- `/NXaperture/TRANSFORMATIONS-group`
- `/NXaperture/@default-attribute`

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/base_classes/NXaperture.nxdl.xml

**NXattenuator**

**Status:**

base class, extends `NXobject`

**Description:**

A device that reduces the intensity of a beam by attenuation.

If uncertain whether to use `NXfilter` (band-pass filter) or `NXattenuator` (reduces beam intensity), then choose `NXattenuator`.

**Symbols:**

No symbol table

**Groups cited:**

`NXoff_geometry, NXtransformations`

**Structure:**

`@default`: (optional) `NX_CHAR`
Declares which child group contains a path leading to a NxData group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

distance: (optional) NX_FLOAT {units=NX_LENGTH}

Distance from sample. Note, it is recommended to use NXtransformations instead.

type: (optional) NX_CHAR

Type or composition of attenuator, e.g. polythene

thickness: (optional) NX_FLOAT {units=NX_LENGTH}

Thickness of attenuator along beam direction

scattering_cross_section: (optional) NX_FLOAT {units=NX_CROSS_SECTION}

Scattering cross section (coherent+incoherent)

absorption_cross_section: (optional) NX_FLOAT {units=NX_CROSS_SECTION}

Absorption cross section

attenuator_transmission: (optional) NX_FLOAT {units=NX_DIMENSIONLESS}

The nominal amount of the beam that gets through (transmitted intensity)/(incident intensity)

status: (optional) NX_CHAR

In or out or moving of the beam

Any of these values: in | out | moving

@time: (optional) NX_DATE_TIME

time stamp for this observation

depends_on: (optional) NX_CHAR

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

The reference point of the attenuator is its center in the x and y axis. The reference point on the z axis is the surface of the attenuator pointing towards the source.

In complex (asymmetric) geometries an NXoff_geometry group can be used to provide an unambiguous reference.

TRANSFORMATIONS: (optional) NXtransformations

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.
**shape**: (optional) *NXoff_geometry*

Shape of this component. Particularly useful to define the origin for position and orientation in non-standard cases.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXattenuator/absorption_cross_section-field
- /NXattenuator/attenuator_transmission-field
- /NXattenuator/depends_on-field
- /NXattenuator/distance-field
- /NXattenuator/scattering_cross_section-field
- /NXattenuator/shape-group
- /NXattenuator/status-field
- /NXattenuator/status@time-attribute
- /NXattenuator/thickness-field
- /NXattenuator/TRANSFORMATIONS-group
- /NXattenuator/type-field
- /NXattenuator@default-attribute

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/base_classes/NXattenuator.nxdl.xml

**NXbeam**

**Status:**

base class, extends *NXobject*

**Description:**

Properties of the neutron or X-ray beam at a given location.

This group is intended to be referenced by beamline component groups within the *NXinstrument* group or by the *NXsample* group. 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.

Note that incident_wavelength and related fields can be a scalar values or arrays, depending on the use case. To support these use cases, the explicit dimensionality of these fields is not specified, but it can be inferred by the presence of and shape of accompanying fields, such as incident_wavelength_weights for a polychromatic beam.

**Symbols:**

These symbols coordinate datasets with the same shape.

- **nP**: Number of scan points.
- **m**: Number of channels in the incident beam spectrum, if known
c: Number of moments representing beam divergence (x, y, xy, etc.)

Groups cited:
NXdata, NXtransformations

Structure:

@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

distance: (optional) NX_FLOAT {units=NX_LENGTH}

Distance from sample. Note, it is recommended to use NXtransformations instead.

incident_energy: (optional) NX_FLOAT (Rank: 1, Dimensions: [m]) {units=NX_ENERGY}

Energy carried by each particle of the beam on entering the beamline component.

final_energy: (optional) NX_FLOAT (Rank: 1, Dimensions: [m]) {units=NX_ENERGY}

Energy carried by each particle of the beam on leaving the beamline component.

energy_transfer: (optional) NX_FLOAT (Rank: 1, Dimensions: [m]) {units=NX_ENERGY}

Change in particle energy caused by the beamline component.

incident_wavelength: (optional) NX_FLOAT {units=NX_WAVELENGTH}

In the case of a monochromatic beam this is the scalar wavelength.

Several other use cases are permitted, depending on the presence or absence of other incident_wavelength_X fields.

In the case of a polychromatic beam this is an array of length m of wavelengths, with the relative weights in incident_wavelength_weights.

In the case of a monochromatic beam that varies shot-to-shot, this is an array of wavelengths, one for each recorded shot. Here, incident_wavelength_weights and incident_wavelength_spread are not set.

In the case of a polychromatic beam that varies shot-to-shot, this is an array of length m with the relative weights in incident_wavelength_weights as a 2D array.

In the case of a polychromatic beam that varies shot-to-shot and where the channels also vary, this is a 2D array of dimensions nP by m (slow to fast) with the relative weights in incident_wavelength_weights as a 2D array.

Note, variants are a good way to represent several of these use cases in a single dataset, e.g. if a calibrated, single-value wavelength value is available along with the original spectrum from which it was calibrated. Wavelength on entering beamline component.

incident_wavelength_weights: (optional) NX_FLOAT

In the case of a polychromatic beam this is an array of length m of the relative weights of the corresponding wavelengths.

In the case of a polychromatic beam that varies shot-to-shot, this is a 2D array of dimensions nP by m (slow to fast) of the relative weights of the corresponding wavelengths in incident_wavelength.
incident_wavelength_spread:  (optional) NX_FLOAT  (Rank: 1, Dimensions: [nP])  
{units=NX_WAVELENGTH}

The wavelength spread FWHM for the corresponding wavelength(s) in incident_wavelength. 

In the case of shot-to-shot variation in the wavelength spread, this is a 2D array of dimension 
nP by m (slow to fast) of the spreads of the corresponding wavelengths in incident_wavelength.

incident_beam_divergence:  (optional) NX_FLOAT  (Rank: 2, Dimensions: [nP, c])  
{units=NX_ANGLE}

Beam crossfire in degrees parallel to the laboratory X axis

The dimension c is a series of moments of that represent the standard uncertainty (e.s.d.) of the 
directions of of the beam. The first and second moments are in the XZ and YZ planes around 
the mean source beam direction, respectively.

Further moments in c characterize co-variance terms, so the next moment is the product of the 
first two, and so on.

extent:  (optional) NX_FLOAT  (Rank: 2, Dimensions: [nP, 2])  
{units=NX_LENGTH}

Size of the beam entering this component. Note this represents a rectangular beam aperture, 
and values represent FWHM

final_wavelength:  (optional) NX_FLOAT  (Rank: 1, Dimensions: [m])  
{units=NX_WAVELENGTH}

Wavelength on leaving beamline component

incident_polization:  (optional) NX_NUMBER  (Rank: 2, Dimensions: [nP, 2])  
{units=NX_ANY}

Polarization vector on entering beamline component

final_polization:  (optional) NX_NUMBER  (Rank: 2, Dimensions: [nP, 2])  
{units=NX_ANY}

Polarization vector on leaving beamline component

incident_polization_stokes:  (optional) NX_NUMBER  (Rank: 2, Dimensions: [nP, 4])  
{units=NX_ANY}

Polarization vector on entering beamline component using Stokes notation

The Stokes parameters are four components labelled I,Q,U,V or S_0,S_1,S_2,S_3. These are 
defined with the standard Nexus coordinate frame unless it is overridden by an NTransformations 
field pointed to by a depends_on attribute. The last component, describing the circular 
polarization state, is positive for a right-hand circular state - that is the electric field vector 
rotates clockwise at the sample and over time when observed from the source.

I (S_0) is the beam intensity (often normalized to 1). Q, U, and V scale linearly with the 
total degree of polarization, and indicate the relative magnitudes of the pure linear and circular 
orientation contributions.

Q (S_1) is linearly polarized along the x axis (Q > 0) or y axis (Q < 0).

U (S_2) is linearly polarized along the x==y axis (U > 0) or the -x==y axis (U < 0).

V (S_3) is circularly polarized. V > 0 when the electric field vector rotates clockwise at the 
sample with respect to time when observed from the source; V < 0 indicates the opposite rota-
tion.

final_polization_stokes:  (optional) NX_NUMBER  (Rank: 2, Dimensions: [nP, 4])  
{units=NX_ANY}

Polarization vector on leaving beamline component using Stokes notation (see incident_polization_stokes).

final_wavelength_spread:  (optional) NX_FLOAT  (Rank: 1, Dimensions: [m])  
{units=NX_WAVELENGTH}

3.3. NeXus Class Definitions 179
Wavelength spread FWHM of beam leaving this component

**final_beam_divergence**: (optional) `NX_FLOAT` (Rank: 2, Dimensions: [nP, 2]) \{units=`NX_ANGLE`\}

Divergence FWHM of beam leaving this component

**flux**: (optional) `NX_FLOAT` (Rank: 1, Dimensions: [nP]) \{units=`NX_FLUX`\}

flux incident on beam plane area

**depends_on**: (optional) `NX_CHAR`

The NeXus coordinate system defines the Z axis to be along the nominal beam direction. This is the same as the McStas coordinate system (see :ref:`Design-CoordinateSystem`). However, the additional transformations needed to represent an altered beam direction can be provided using this depends_on field that contains the path to a NXtransformations group. This could represent redirection of the beam, or a refined beam direction.

**DATA**: (optional) `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.

**TRANSFORMATIONS**: (optional) `NXtransformations`

Direction (and location) for the beam. The location of the beam can be given by any point which it passes through as its offset attribute.

**DIRECTION**: (optional) `NX_NUMBER` \{units=`NX_TRANSFORMATION`\}

Direction of beam vector, its value is ignored. If missing, then the beam direction is defined as [0,0,1] and passes through the origin

@**transformation_type**: (optional) `NX_CHAR`

Obligatory value: translation

@**vector**: (optional) `NX_NUMBER`

Three values that define the direction of beam vector

@**offset**: (optional) `NX_NUMBER`

Three values that define the location of a point through which the beam passes

@**depends_on**: (optional) `NX_CHAR`

Points to the path to a field defining the location on which this depends or the string “.” for origin.

**reference_plane**: (optional) `NX_NUMBER` \{units=`NX_TRANSFORMATION`\}

Direction of normal to reference plane used to measure azimuth relative to the beam, its value is ignored. This also defines the parallel and perpendicular components of the beam’s polarization. If missing, then the reference plane normal is defined as [0,1,0] and passes through the origin

@**transformation_type**: (optional) `NX_CHAR`

Obligatory value: translation

@**vector**: (optional) `NX_NUMBER`

Three values that define the direction of reference plane normal

@**offset**: (optional) `NX_NUMBER`

Not required as beam direction offset locates the plane
@depends_on: (optional) *NX_CHAR*

Points to the path to a field defining the location on which this depends or the string "." for origin.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXbeam/DATA-group
- /NXbeam/depends_on-field
- /NXbeam/distance-field
- /NXbeam/energy_transfer-field
- /NXbeam/extent-field
- /NXbeam/final_beam_divergence-field
- /NXbeam/final_energy-field
- /NXbeam/final_polarization-field
- /NXbeam/final_polarization_stokes-field
- /NXbeam/final_wavelength-field
- /NXbeam/final_wavelength_spread-field
- /NXbeam/flux-field
- /NXbeam/incident_beam_divergence-field
- /NXbeam/incident_energy-field
- /NXbeam/incident_polarization-field
- /NXbeam/incident_polarization_stokes-field
- /NXbeam/incident_wavelength-field
- /NXbeam/incident_wavelength_spread-field
- /NXbeam/incident_wavelength_weights-field
- /NXbeam/TRANSFORMATIONS-group
- /NXbeam/TRANSFORMATIONS/DIRECTION-field
- /NXbeam/TRANSFORMATIONS/DIRECTION@depends_on-attribute
- /NXbeam/TRANSFORMATIONS/DIRECTION@offset-attribute
- /NXbeam/TRANSFORMATIONS/DIRECTION@transformation_type-attribute
- /NXbeam/TRANSFORMATIONS/DIRECTION@vector-attribute
- /NXbeam/TRANSFORMATIONS/reference_plane-field
- /NXbeam/TRANSFORMATIONS/reference_plane@depends_on-attribute
- /NXbeam/TRANSFORMATIONS/reference_plane@offset-attribute
- /NXbeam/TRANSFORMATIONS/reference_plane@transformation_type-attribute
- /NXbeam/TRANSFORMATIONS/reference_plane@vector-attribute
NXbeam_stop

Status:

base class, extends NXobject

Description:

A device that blocks the beam completely, usually to protect a detector.

Beamstops and their positions are important for SANS and SAXS experiments.

Symbols:

No symbol table

Groups cited:

NXcylindrical_geometry, NXgeometry, NXoff_geometry, NXtransformations

Structure:

@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

description: (optional) NX_CHAR

description of beamstop

Any of these values: circular | rectangular

size: (optional) NX_FLOAT {units=NX_LENGTH}

Size of beamstop. If this is not sufficient to describe the beam stop use NXoff_geometry instead.

x: (optional) NX_FLOAT {units=NX_LENGTH}

x position of the beamstop in relation to the detector. Note, it is recommended to use NXtransformations instead.

y: (optional) NX_FLOAT {units=NX_LENGTH}

y position of the beamstop in relation to the detector. Note, it is recommended to use NXtransformations instead.

distance_to_detector: (optional) NX_FLOAT {units=NX_LENGTH}

distance of the beamstop to the detector. Note, it is recommended to use NXtransformations instead.

status: (optional) NX_CHAR

Any of these values: in | out

depends_on: (optional) NX_CHAR
NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

The reference point of the beam stop is its center in the x and y axis. The reference point on the z axis is the surface of the beam stop pointing towards the source.

**GEOMETRY**: (optional) **NXgeometry**

**DEPRECATED**: Use the field depends_on and **NXtransformations** to position the beamstop and **NXoff_geometry** to describe its shape instead engineering shape, orientation and position of the beam stop.

**OFF_GEOMETRY**: (optional) **NXoff_geometry**

This group describes the shape of the beam line component

**CYLINDRICAL_GEOMETRY**: (optional) **NXcylindrical_geometry**

This group is an alternative to NXoff_geometry for describing the shape of the beam stop.

**TRANSFORMATIONS**: (optional) **NXtransformations**

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXbeam_stop/CYLINDRICAL_GEOMETRY-group
- /NXbeam_stop/depends_on-field
- /NXbeam_stop/description-field
- /NXbeam_stop/distance_to_detector-field
- /NXbeam_stop/GEOMETRY-group
- /NXbeam_stop/OFF_GEOMETRY-group
- /NXbeam_stop/size-field
- /NXbeam_stop/status-field
- /NXbeam_stop/TRANSFORMATIONS-group
- /NXbeam_stop/x-field
- /NXbeam_stop/y-field
- /NXbeam_stop@default-attribute
NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXbeam_stop.nxdl.xml

NXbending_magnet

Status:
base class, extends NXobject

Description:
A bending magnet

Symbols:
No symbol table

Groups cited:
NXdata, NXgeometry, NXoff_geometry, NXtransformations

Structure:

@default: (optional) NX_CHAR
Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

critical_energy: (optional) NX_FLOAT {units=NX_ENERGY}
bending_radius: (optional) NX_FLOAT {units=NX_LENGTH}
magnetic_field: (optional) NX_FLOAT {units=NX_CURRENT}

strength of magnetic field of dipole magnets

accepted_photon_beam_divergence: (optional) NX_FLOAT {units=NX_LENGTH}
An array of four numbers giving X+, X-, Y+ and Y- half divergence

source_distance_x: (optional) NX_FLOAT {units=NX_LENGTH}
Distance of source point from particle beam waist in X (horizontal) direction. Note, it is recommended to use NXtransformations instead to place component.

source_distance_y: (optional) NX_FLOAT {units=NX_LENGTH}
Distance of source point from particle beam waist in Y (vertical) direction. Note, it is recommended to use NXtransformations instead to place component.

divergence_x_plus: (optional) NX_FLOAT {units=NX_ANGLE}
Accepted photon beam divergence in X+ (horizontal outboard) direction. Note that divergence_x_plus+divergence_x_minus is the total horizontal beam divergence.

divergence_x_minus: (optional) NX_FLOAT {units=NX_ANGLE}
Accepted photon beam divergence in X- (horizontal inboard) direction. Note that divergence_x_plus+divergence_x_minus is the total horizontal beam divergence.

divergence_y_plus: (optional) NX_FLOAT {units=NX_ANGLE}
Accepted photon beam divergence in Y+ (vertical upward) direction. Note that divergence_y_plus+divergence_y_minus is the total vertical beam divergence.
**divergence_y_minus**: (optional) `NX_FLOAT {units=NX_ANGLE}`

Accepted photon beam divergence in Y- (vertical downward) direction. Note that divergence_y_plus+divergence_y_minus is the total vertical beam divergence.

**depends_on**: (optional) `NX_CHAR`

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

**spectrum**: (optional) `NXdata`

bending magnet spectrum

**GEOMETRY**: (optional) `NXgeometry`

DEPRECATED: Use the field depends_on and `NXtransformations` to position the bending magnet and NXoff_geometry to describe its shape instead

“Engineering” position of bending magnet

**OFF_GEOMETRY**: (optional) `NXoff_geometry`

This group describes the shape of the beam line component

**TRANSFORMATIONS**: (optional) `NXtransformations`

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXbending_magnet/accepted_photon_beam_divergence-field`
- `/NXbending_magnet/bending_radius-field`
- `/NXbending_magnet/critical_energy-field`
- `/NXbending_magnet/depends_on-field`
- `/NXbending_magnet/divergence_x_minus-field`
- `/NXbending_magnet/divergence_x_plus-field`
- `/NXbending_magnet/divergence_y_minus-field`
- `/NXbending_magnet/divergence_y_plus-field`
- `/NXbending_magnet/GEOMETRY-group`
- `/NXbending_magnet/magnetic_field-field`
- `/NXbending_magnet/OF.getLocalDataF(r).GEOMETRY-group`
- `/NXbending_magnet/source_distance_x-field`
- `/NXbending_magnet/source_distance_y-field`
- `/NXbending_magnet/spectrum-group`
NXcapillary

Status:
base class, extends NXobject

Description:
A capillary lens to focus the X-ray beam.
Based on information provided by Gerd Wellenreuther (DESY).

Symbols:
No symbol table

Groups cited:
NXdata, NXtransformations

Structure:

@default: (optional) NX_CHAR
Declares which child group contains a path leading to a NXdata group.
It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

type: (optional) NX_CHAR
Type of the capillary
Any of these values:
• single_bounce
• polycapillary
• conical_capillary

manufacturer: (optional) NX_CHAR
The manufacturer of the capillary. This is actually important as it may have an impact on performance.

maximum_incident_angle: (optional) NX_FLOAT {units=NX_ANGLE}

accepting_aperture: (optional) NX_FLOAT {units=NX_ANGLE}

working_distance: (optional) NX_FLOAT {units=NX_LENGTH}

focal_size: (optional) NX_FLOAT
The focal size in FWHM

depends_on: (optional) NX_CHAR
NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

**gain:** (optional) **NXdata**

The gain of the capillary as a function of energy

**transmission:** (optional) **NXdata**

The transmission of the capillary as a function of energy

**TRANSFORMATIONS:** (optional) **NXtransformations**

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXcapillary/accepting_aperture-field`
- `/NXcapillary/depends_on-field`
- `/NXcapillary/focal_size-field`
- `/NXcapillary/gain-group`
- `/NXcapillary/manufacturer-field`
- `/NXcapillary/maximum_incident_angle-field`
- `/NXcapillary/TRANSFORMATIONS-group`
- `/NXcapillary/transmission-group`
- `/NXcapillary/type-field`
- `/NXcapillary/working_distance-field`
- `/NXcapillary@default-attribute`

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/base_classes/NXcapillary.nxdl.xml

**NXcite**

**Status:**

- base class, extends **NXobject**

**Description:**

A literature reference

Definition to include references for example for detectors, manuals, instruments, acquisition or analysis software used.

The idea would be to include this in the relevant NeXus object: **NXdetector** for detectors, **NXinstrument** for instruments, etc.
Symbols:

No symbol table

Groups cited:

none

Structure:

@default: (optional) $NX\_CHAR$

Declares which child group contains a path leading to a $NXdata$ group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

description: (optional) $NX\_CHAR$

This should describe the reason for including this reference. For example: The dataset in this group was normalised using the method which is described in detail in this reference.

url: (optional) $NX\_CHAR$

URL referencing the document or data.

doi: (optional) $NX\_CHAR$

DOI referencing the document or data.

endnote: (optional) $NX\_CHAR$

Bibliographic reference data in EndNote format.

bibtex: (optional) $NX\_CHAR$

Bibliographic reference data in BibTeX format.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXcite/bibtex-field
- /NXcite/description-field
- /NXcite/doi-field
- /NXcite/endnote-field
- /NXcite/url-field
- /NXcite@default-attribute

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/base_classes/NXcite.nxdl.xml
**NXcollection**

**Status:**
base class, extends *NXobject*

**Description:**
An unvalidated set of terms, such as the description of a beam line.

Use *NXcollection* to gather together any set of terms. The original suggestion is to use this as a container class for the description of a beamline.

For NeXus validation, *NXcollection* will always generate a warning since it is always an optional group. Anything (groups, fields, or attributes) placed in an *NXcollection* group will not be validated.

**Symbols:**
No symbol table

**Groups cited:**
none

**Structure:**

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/base_classes/NXcollection.nxdl.xml

---

**NXcollimator**

**Status:**
base class, extends *NXobject*

**Description:**
A beamline collimator.

**Symbols:**
No symbol table

**Groups cited:**
*NXgeometry*, *NXlog*, *NXoff_geometry*, *NXtransformations*

**Structure:**

@default: (optional) *NX_CHAR*

Declares which child group contains a path leading to a *NXdata* group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

**type:** (optional) *NX_CHAR*

Any of these values: Soller | radial | oscillating | honeycomb

**soller_angle:** (optional) *NX_FLOAT* {units=*NX_ANGLE*}

Angular divergence of Soller collimator

**divergence_x:** (optional) *NX_FLOAT* {units=*NX_ANGLE*}

divergence of collimator in local x direction

---

3.3. NeXus Class Definitions
**divergence_y**: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

  divergence of collimator in local y direction

**frequency**: (optional) *NX_FLOAT* {units=*NX_FREQUENCY*}

  Frequency of oscillating collimator

**blade_thickness**: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

  blade thickness

**blade_spacing**: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

  blade spacing

**absorbing_material**: (optional) *NX_CHAR*

  name of absorbing material

**transmitting_material**: (optional) *NX_CHAR*

  name of transmitting material

**depends_on**: (optional) *NX_CHAR*

  NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

  Assuming a collimator with a “flat” entry surface, the reference plane is the plane which contains this surface. The reference point of the collimator in the x and y axis is the centre of the collimator entry surface on that plane. The reference plane is orthogonal to the z axis and the location of this plane is the reference point on the z axis. The collimator faces negative z values.

**GEOMETRY**: (optional) *NXgeometry*

  **DEPRECATED**: Use the field depends_on and *NXtransformations* to position the collimator and *NXoff_geometry* to describe its shape instead

  position, shape and size

**frequency_log**: (optional) *NXlog*

  Log of frequency

**OFF_GEOMETRY**: (optional) *NXoffGeometry*

  This group describes the shape of the beam line component

**TRANSFORMATIONS**: (optional) *NXtransformations*

  This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXcollimator/absorbing_material-field
- /NXcollimator/blade_spacing-field
- /NXcollimator/blade_thickness-field
- /NXcollimator/depends_on-field
- /NXcollimator/divergence_x-field
- /NXcollimator/divergence_y-field
- /NXcollimator/frequency-field
- /NXcollimator/frequency_log-group
- /NXcollimator/GEOMETRY-group
- /NXcollimator/OFF_GEOMETRY-group
- /NXcollimator/soller_angle-field
- /NXcollimator/TRANSFORMATIONS-group
- /NXcollimator/transmitting_material-field
- /NXcollimator/type-field
- /NXcollimator@default-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXcollimator.nxdl.xml

NXcrystal

Status:
base class, extends NXobject

Description:
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.

Symbols:
These symbols will be used below to coordinate dimensions with the same lengths.

- n_comp: number of different unit cells to be described
- i: number of wavelengths

Groups cited:
NXdata, NXgeometry, NXlog, NXoff_geometry, NXshape, NXtransformations

Structure:
@default: (optional) **NX_CHAR**

Declares which child group contains a path leading to a **NXdata** group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See [https://www.nexusformat.org/2014_How_to_find_default_data.html](https://www.nexusformat.org/2014_How_to_find_default_data.html) for a summary of the discussion.

**usage:** (optional) **NX_CHAR**

How this crystal is used. Choices are in the list.

Any of these values:

- **Bragg**: reflection geometry
- **Laue**: 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. See, for example: [http://www.iucr.org/__data/iucr/cif/standard/cifstd15.html](http://www.iucr.org/__data/iucr/cif/standard/cifstd15.html) or [http://www.cas.org/training/stneasytips/subinforformula1.html](http://www.cas.org/training/stneasytips/subinforformula1.html).

**type:** (optional) **NX_CHAR**

Type or material of monochromating substance. Chemical formula can be specified separately. Use the “reflection” field to indicate the (hkl) orientation. Use the “d_spacing” field to record the lattice plane spacing.

This field was changed (2010-11-17) from an enumeration to a string since common usage showed a wider variety of use than a simple list. These are the items in the list at the time of the change: **PG** (Highly Oriented Pyrolytic Graphite) | **Ge** | **Si** | **Cu** | **Fe3Si** | **CoFe** | **Cu2MnAl** (Heusler) | **Multilayer** | **Diamond**.

**chemical_formula:** (optional) **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.

**order_no**: (optional) *NX_INT*

A number which describes if this is the first, second, ..., $n^{th}$ crystal in a multi crystal monochromator

**cut_angle**: (optional) *NX_FLOAT* \{units=*NX_ANGLE*\}

Cut angle of reflecting Bragg plane and plane of crystal surface

**space_group**: (optional) *NX_CHAR*

Space group of crystal structure

**unit_cell**: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [n_comp, 6]) \{units=*NX_LENGTH*\}

Unit cell parameters (lengths and angles)

**unit_cell_a**: (optional) *NX_FLOAT* \{units=*NX_LENGTH*\}

Unit cell lattice parameter: length of side a

**unit_cell_b**: (optional) *NX_FLOAT* \{units=*NX_LENGTH*\}

Unit cell lattice parameter: length of side b

**unit_cell_c**: (optional) *NX_FLOAT* \{units=*NX_LENGTH*\}

Unit cell lattice parameter: length of side c

**unit_cell_alpha**: (optional) *NX_FLOAT* \{units=*NX_ANGLE*\}

Unit cell lattice parameter: angle alpha

**unit_cell_beta**: (optional) *NX_FLOAT* \{units=*NX_ANGLE*\}

Unit cell lattice parameter: angle beta

**unit_cell_gamma**: (optional) *NX_FLOAT* \{units=*NX_ANGLE*\}

Unit cell lattice parameter: angle gamma

**unit_cell_volume**: (optional) *NX_FLOAT* \{units=*NX_VOLUME*\}

Volume of the unit cell

**orientation_matrix**: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [3, 3])


**wavelength**: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [i]) \{units=*NX_WAVELENGTH*\}

Optimum diffracted wavelength

**d_spacing**: (optional) *NX_FLOAT* \{units=*NX_LENGTH*\}

Spacing between crystal planes of the reflection

**scattering_vector**: (optional) *NX_FLOAT* \{units=*NX_WAVENUMBER*\}

Scattering vector, Q, of nominal reflection

**reflection**: (optional) *NX_INT* (Rank: 1, Dimensions: [3]) \{units=*NX_UNITLESS*\}

Miller indices (hkl) values of nominal reflection

**thickness**: (optional) *NX_FLOAT* \{units=*NX_LENGTH*\}

Thickness of the crystal. (Required for Laue orientations - see “usage” field)
density: (optional) $NX_NUMBER \ {units=NX\_MASS\_DENSITY}$
    mass density of the crystal

segment_width: (optional) $NX\_FLOAT \ {units=NX\_LENGTH}$
    Horizontal width of individual segment

segment_height: (optional) $NX\_FLOAT \ {units=NX\_LENGTH}$
    Vertical height of individual segment

segment_thickness: (optional) $NX\_FLOAT \ {units=NX\_LENGTH}$
    Thickness of individual segment

segment_gap: (optional) $NX\_FLOAT \ {units=NX\_LENGTH}$
    Typical gap between adjacent segments

segment_columns: (optional) $NX\_FLOAT \ {units=NX\_LENGTH}$
    number of segment columns in horizontal direction

segment_rows: (optional) $NX\_FLOAT \ {units=NX\_LENGTH}$
    number of segment rows in vertical direction

mosaic_horizontal: (optional) $NX\_FLOAT \ {units=NX\_ANGLE}$
    horizontal mosaic Full Width Half Maximum

mosaic_vertical: (optional) $NX\_FLOAT \ {units=NX\_ANGLE}$
    vertical mosaic Full Width Half Maximum

curvature_horizontal: (optional) $NX\_FLOAT \ {units=NX\_ANGLE}$
    Horizontal curvature of focusing crystal

curvature_vertical: (optional) $NX\_FLOAT \ {units=NX\_ANGLE}$
    Vertical curvature of focusing crystal

is_cylindrical: (optional) $NX\_BOOLEAN$
    Is this crystal bent cylindrically?

cylindrical_orientation_angle: (optional) $NX\_NUMBER \ {units=NX\_ANGLE}$
    If cylindrical: cylinder orientation angle

polar_angle: (optional) $NX\_FLOAT \ {Rank:1, Dimensions: [i]} \ {units=NX\_ANGLE}$
    Polar (scattering) angle at which crystal assembly is positioned. Note: some instrument geometries call this term 2theta. Note: it is recommended to use NXtransformations instead.

azimuthal_angle: (optional) $NX\_FLOAT \ {Rank:1, Dimensions: [i]} \ {units=NX\_ANGLE}$
    Azimuthal angle at which crystal assembly is positioned. Note: it is recommended to use NXtransformations instead.

bragg_angle: (optional) $NX\_FLOAT \ {Rank:1, Dimensions: [i]} \ {units=NX\_ANGLE}$
    Bragg angle of nominal reflection

temperature: (optional) $NX\_FLOAT \ {units=NX\_TEMPERATURE}$
    average/nominal crystal temperature

temperature_coefficient: (optional) $NX\_FLOAT \ {units=NX\_ANY}$
how lattice parameter changes with temperature

**depends_on:** (optional) *NX_CHAR*

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string "." if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

**GEOMETRY:** (optional) *NXgeometry*

**DEPRECATED:** Use the field depends_on and *NXtransformations* to position the crystal and NXoff_geometry to describe its shape instead

Position of crystal

**temperature_log:** (optional) *NXlog*

log file of crystal temperature

**reflectivity:** (optional) *NXdata*

crystal reflectivity versus wavelength

**transmission:** (optional) *NXdata*

crystal transmission versus wavelength

**shape:** (optional) *NXshape*

**DEPRECATED:** Use NXoff_geometry instead to describe the shape of the monochromator

A NXshape group describing the shape of the crystal arrangement

**OFF_GEOMETRY:** (optional) *NXoff_geometry*

This group describes the shape of the beam line component

**TRANSFORMATIONS:** (optional) *NXtransformations*

Transformations used by this component to define its position and orientation.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXcrystal/azimuthal_angle-field
- /NXcrystal/bragg_angle-field
- /NXcrystal/chemical_formula-field
- /NXcrystal/curvature_horizontal-field
- /NXcrystal/curvature_vertical-field
- /NXcrystal/cut_angle-field
- /NXcrystal/cylindrical_orientation_angle-field
- /NXcrystal/d_spacing-field
- /NXcrystal/density-field
- /NXcrystal/depends_on-field
• /NXcrystal/GEOMETRY-group
• /NXcrystal/is_cylindrical-field
• /NXcrystal/mosaic_horizontal-field
• /NXcrystal/mosaic_vertical-field
• /NXcrystal/OFF_GEOMETRY-group
• /NXcrystal/order_no-field
• /NXcrystal/orientation_matrix-field
• /NXcrystal/polar_angle-field
• /NXcrystal/reflection-field
• /NXcrystal/reflectivity-group
• /NXcrystal/scattering_vector-field
• /NXcrystal/segment_columns-field
• /NXcrystal/segment_gap-field
• /NXcrystal/segment_height-field
• /NXcrystal/segment_rows-field
• /NXcrystal/segment_thickness-field
• /NXcrystal/segment_width-field
• /NXcrystal/shape-group
• /NXcrystal/space_group-field
• /NXcrystal/temperature-field
• /NXcrystal/temperature_coefficient-field
• /NXcrystal/temperature_log-group
• /NXcrystal/thickness-field
• /NXcrystal/TRANSFORMATIONS-group
• /NXcrystal/transmission-group
• /NXcrystal/type-field
• /NXcrystal/unit_cell-field
• /NXcrystal/unit_cell_a-field
• /NXcrystal/unit_cell_alpha-field
• /NXcrystal/unit_cell_b-field
• /NXcrystal/unit_cell_beta-field
• /NXcrystal/unit_cell_c-field
• /NXcrystal/unit_cell_gamma-field
• /NXcrystal/unit_cell_volume-field
• /NXcrystal/usage-field
• /NXcrystal/wavelength-field
NXcylindrical_geometry

Status:
base class, extends NXobject

Description:
Geometry description for cylindrical shapes. This class can be used in place of NXoff_geometry when an exact representation for cylinders is preferred. For example, for Helium-tube, neutron detectors. It can be used to describe the shape of any beamline component, including detectors. In the case of detectors it can be used to define the shape of a single pixel, or, if the pixel shapes are non-uniform, to describe the shape of the whole detector.

Symbols:
These symbols will be used below.
i: number of vertices required to define all cylinders in the shape
j: number of cylinders in the shape
k: number cylinders which are detectors

Groups cited:
none

Structure:
@default: (optional) NX_CHAR
Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

vertices: (optional) NX_NUMBER (Rank: 2, Dimensions: [i, 3]) {units=NX_LENGTH}
List of x,y,z coordinates for vertices. The origin of the coordinates is the position of the parent component, for example the NXdetector which the geometry describes. If the shape describes a single pixel for a detector with uniform pixel shape then the origin is the position of each pixel as described by the x/y/z_pixel_offset datasets in NXdetector.

cylinders: (optional) NX_INT (Rank: 2, Dimensions: [j, 3])
List of indices of vertices in the vertices dataset to form each cylinder. Each cylinder is described by three vertices A, B, C. First vertex A lies on the cylinder axis and circular face, second point B on edge of the same face as A, and third point C at the other face and on axis.

detector_number: (optional) NX_INT (Rank: 1, Dimensions: [k])
Maps cylinders in cylinder, by index, with a detector id.
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXcylindrical Geometry/cylinders-field
- /NXcylindrical Geometry/detector_number-field
- /NXcylindrical Geometry/vertices-field
- /NXcylindrical Geometry@default-attribute

NXDL Source:

NXdata

Status:
base class, extends NXobject

Description:

NXdata describes the plottable data and related dimension scales.

It is strongly recommended that there is at least one NXdata group in each NXentry group. Note that the fields named AXISNAME and DATA can be defined with different names. (Upper case is used to indicate that the actual name is left to the user.) The signal and axes attributes of the data group define which items are plottable data and which are dimension scales, respectively.

NXdata is used to implement one of the basic motivations in NeXus, to provide a default plot for the data of this NXentry. The actual data might be stored in another group and (hard) linked to the NXdata group.

- Each NXdata group will define one field as the default plottable data. The value of the signal attribute names this field. Additional fields may be used to describe the dimension scales and uncertainties. The auxiliary_signals attribute is a list of the other fields to be plotted with the signal data.
- The plottable data may be of arbitrary rank up to a maximum of NX_MAXRANK=32 (for compatibility with backend file formats).
- The plottable data will be named as the value of the group signal attribute, such as:

```plaintext
data:NXdata
  @signal = "counts"
  @axes = "mr"
  @mr_indices = 0
  counts: float[100] --> the default dependent data
  mr: float[100] --> the default independent data
```

The field named in the signal attribute must exist, either directly as a NeXus field or defined through a link.

- The group axes attribute will name the dimension scale associated with the plottable data.

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, such as scattering angle or energy transfer.
The preferred method to associate each data dimension with its respective dimension scale is to specify the field name of each dimension scale in the group `axes` attribute as a string list. Here is an example for a 2-D data set `data` plotted against `time`, and `pressure`. (An additional `temperature` data set is provided and could be selected as an alternate for the `pressure` axis):

```plaintext
data_2d:NXdata
  @signal="data"
  @axes="["time", "pressure"]"
  @pressure_indices=1
  @temperature_indices=1
  @time_indices=0
  data: float[1000,20]
  pressure: float[20]
  temperature: float[20]
  time: float[1000]
```

### Old methods to identify the plottable data

There are two older methods of associating each data dimension to its respective dimension scale. Both are now out of date and should not be used when writing new data files. However, client software should expect to see data files written with any of these methods.

- One method uses the `axes` attribute to specify the names of each `dimension scale`.
- The oldest method uses the `axis` attribute on each `dimension scale` to identify with an integer the axis whose value is the number of the dimension.

Each axis of the plot may be labeled with information from the dimension scale for that axis. The optional `@long_name` attribute is provided as the axis label default. If `@long_name` is not defined, then use the name of the dimension scale. A `@units` attribute, if available, may be added to the axis label for further description. See the section `NeXus Data Units` for more information.

The optional `title` field, if available, provides a suggested title for the plot. If no `title` field is found in the `NXdata` group, look for a `title` field in the parent `NXentry` group, with a fallback to displaying the path to the `NXdata` group.

NeXus is about how to find and annotate the data to be plotted but not to describe how the data is to be plotted. (https://www.nexusformat.org/NIAC2018Minutes.html#nxdata-plottype–attribute)

**Symbols:**

These symbols will be used below to coordinate fields with the same shape.

- **dataRank**: rank of the `DATA` field
- n: length of the `AXISNAME` field
- nx: length of the `x` field
- ny: length of the `y` field
- nz: length of the `z` field

**Groups cited:**  
none

**Structure:**

- `@auxiliary_signals`: (optional) `NX_CHAR`
Array of strings holding the names of additional signals to be plotted with the default signal (specified by the signal attribute). Each auxiliary signal needs to be of the same shape as the default signal.

@signal: (optional) NX_CHAR
Declares which NeXus field is the default. The value is the name of the NeXus field to be plotted. (The value names an existing child of this group. The child group must itself be a NeXus group.)

It is recommended (as of NIAC2014) to use this attribute rather than adding a signal attribute to the field. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

@axes: (optional) NX_CHAR
String array that defines the independent data fields used in the default plot for all of the dimensions of the signal field (the signal field is the field in this group that is named by the signal attribute of this group). One entry is provided for every dimension in the signal field.

The field(s) named as values (known as “axes”) of this attribute must exist. An axis slice is specified using a field named AXISNAME_indices as described below (where the text shown here as AXISNAME is to be replaced by the actual field name).

When no default axis is available for a particular dimension of the plottable data, use a “.” in that position. Such as:

```plaintext
@axes=["time", ".", "."]
```

Since there are three items in the list, the signal field must must be a three-dimensional array (rank=3). The first dimension is described by the values of a one-dimensional array named time while the other two dimensions have no fields to be used as dimension scales.

See examples provided on the NeXus wiki: https://www.nexusformat.org/2014_axes_and_uncertainties.html

If there are no axes at all (such as with a stack of images), the axes attribute can be omitted.

@AXISNAME_indices: (optional) NX_INT
Each AXISNAME_indices attribute indicates the dependency relationship of the AXISNAME field (where AXISNAME is the name of a field that exists in this NXdata group) with one or more dimensions of the plottable data.

Integer array that defines the indices of the signal field (that field will be a multidimensional array) which need to be used in the AXISNAME field in order to reference the corresponding axis value.

The first index of an array is 0 (zero).

Here, AXISNAME is to be replaced by the name of each field described in the axes attribute. An example with 2-D data, \( d(t, P) \), will illustrate:

```plaintext
data_2d:NXdata
@signal="data"
@axes=["time", "pressure"]
@time_indices=0
@pressure_indices=1
data: float[1000,20]
time: float[1000]
pressure: float[20]
```
This attribute is to be provided in all situations. However, if the indices attributes are missing (such as for data files written before this specification), file readers are encouraged to make their best efforts to plot the data. Thus the implementation of the `AXISNAME_indices` attribute is based on the model of “strict writer, liberal reader”.

**Note:** Attributes potentially containing multiple values (axes and _indices) are to be written as string or integer arrays, to avoid string parsing in reading applications.

### AXISNAME: (optional) `NX_NUMBER` (Rank: 1, Dimensions: [n])

Dimension scale defining an axis of the data. Client is responsible for defining the dimensions of the data. The name of this field may be changed to fit the circumstances. Standard NeXus client tools will use the attributes to determine how to use this field.

- `@long_name`: (optional) `NX_CHAR`
  - Axis label
- `@distribution`: (optional) `NX_BOOLEAN`
  - 0|false: single value, 1|true: multiple values
- `@first_good`: (optional) `NX_INT`
  - Index of first good value
- `@last_good`: (optional) `NX_INT`
  - Index of last good value
- `@axis`: (optional) `NX_POSINT`
  - Index (positive integer) identifying this specific set of numbers.

N.B. The `axis` attribute is the old way of designating a link. Do not use the `axes` attribute with the `axis` attribute. The `axes group` attribute is now preferred.

### FIELDNAME_errors: (optional) `NX_NUMBER`

“Errors” (meaning uncertainties or standard deviations) associated with any field named `FIELDNAME` in this `NXdata` group (e.g. an axis, signal or auxiliary signal).

The dimensions of the `FIELDNAME_errors` field must match the dimensions of the `FIELDNAME` field.

### DATA: (optional) `NX_NUMBER` (Rank: dataRank)

This field contains the data values to be used as the NeXus plottable data. Client is responsible for defining the dimensions of the data. The name of this field may be changed to fit the circumstances. Standard NeXus client tools will use the attributes to determine how to use this field.

- `@signal`: (optional) `NX_POSINT`
  - Plottable (independent) axis, indicate index number. Only one field in a `NXdata` group may have the `signal=1` attribute. Do not use the `signal` attribute with the `axis` attribute.
- `@axes`: (optional) `NX_CHAR`
  - Defines the names of the dimension scales (independent axes) for this data set as a colon-delimited array. NOTE: The `axes` attribute is the preferred method of designating a link. Do not use the `axes` attribute with the `axis` attribute.
@long_name: (optional) NX_CHAR

data label

errors: (optional) NX_NUMBER (Rank: dataRank)

DEPRECATED: Use DATA_errors instead (NIAC2018)

Standard deviations of data values - the data array is identified by the group attribute signal. The errors array must have the same dimensions as DATA. Client is responsible for defining the dimensions of the data.

scaling_factor: (optional) 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: (optional) NX_FLOAT

An optional offset to apply to the values in data.

title: (optional) NX_CHAR

Title for the plot.

x: (optional) NX_FLOAT (Rank: 1, Dimensions: [nx]) {units=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.

y: (optional) NX_FLOAT (Rank: 1, Dimensions: [ny]) {units=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.

z: (optional) NX_FLOAT (Rank: 1, Dimensions: [nz]) {units=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.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXdata/AXISNAME-field
- /NXdata/AXISNAME@axis-attribute
- /NXdata/AXISNAME@distribution-attribute
- /NXdata/AXISNAME@first_good-attribute
- /NXdata/AXISNAME@last_good-attribute
- /NXdata/AXISNAME@long_name-attribute
- /NXdata/DATA-field
- /NXdata/DATA@axes-attribute
- /NXdata/DATA@long_name-attribute
- /NXdata/DATA@signal-attribute
- /NXdata/errors-field
• /NXdata/FIELDNAME_errors-field
• /NXdata/offset-field
• /NXdata/scaling_factor-field
• /NXdata/title-field
• /NXdata/x-field
• /NXdata/y-field
• /NXdata/z-field
• /NXdata@auxiliary_signals-attribute
• /NXdata@axes-attribute
• /NXdata@AXISNAME_indices-attribute
• /NXdata@signal-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXdata.nxdl.xml

NXdetector

Status:
base class, extends NXobject

Description:
A detector, detector bank, or multidetector.

Symbols:
These symbols will be used below to illustrate the coordination of the rank and sizes of datasets and the preferred ordering of the dimensions. Each of these are optional (so the rank of the datasets will vary according to the situation) and the general ordering principle is slowest to fastest. The type of each dimension should follow the order of scan points, detector output (e.g. pixels), then time-of-flight (i.e. spectroscopy, spectrometry). Note that the output of a detector is not limited to single values (0D), lists (1D) and images (2), but three or higher dimensional arrays can be produced by a detector at each trigger.

nP: number of scan points (only present in scanning measurements)
i: number of detector pixels in the first (slowest) direction
j: number of detector pixels in the second (faster) direction
tof: number of bins in the time-of-flight histogram

Groups cited:
NXcollection, NXcylindrical_geometry, NXdata, NXdetector_module, NXgeometry, NXnote, NXoff_geometry, NXtransformations

Structure:
@default: (optional) NX_CHAR
Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

3.3. NeXus Class Definitions
**time_of_flight**: (optional) `NX_FLOAT` (Rank: 1, Dimensions: [tof+1]) {units=`NX_TIME_OF_FLIGHT`}

Total time of flight

@axis: (optional) `NX_POSINT`

Obligatory value: 3

@primary: (optional) `NX_POSINT`

Obligatory value: 1

@long_name: (optional) `NX_CHAR`

Total time of flight

**raw_time_of_flight**: (optional) `NX_INT` (Rank: 1, Dimensions: [tof+1]) {units=`NX_PULSES`}

In DAQ clock pulses

@frequency: (optional) `NX_NUMBER`

Clock frequency in Hz

**detector_number**: (optional) `NX_INT`

Identifier for detector (pixels) Can be multidimensional, if needed

**data**: (optional) `NX_NUMBER` (Rank: 4, Dimensions: [nP, i, j, tof]) {units=`NX_ANY`}

Data values from the detector. The rank and dimension ordering should follow a principle of slowest to fastest measurement axes and may be explicitly specified in application definitions.

Mechanical scanning of objects (e.g. sample position/angle, incident beam energy, etc) tends to be the slowest part of an experiment and so any such scan axes should be allocated to the first dimensions of the array. Note that in some cases it may be useful to represent a 2D set of scan points as a single scan-axis in the data array, especially if the scan pattern doesn’t fit a rectangular array nicely. Repetition of an experiment in a time series tends to be used similar to a slow scan axis and so will often be in the first dimension of the data array.

The next fastest axes are typically the readout of the detector. A point detector will not add any dimensions (as it is just a single value per scan point) to the data array, a strip detector will add one dimension, an imaging detector will add two dimensions (e.g. X, Y axes) and detectors outputting higher dimensional data will add the corresponding number of dimensions. Note that the detector dimensions don’t necessarily have to be written in order of the actual readout speeds - the slowest to fastest rule principle is only a guide.

Finally, detectors that operate in a time-of-flight mode, such as a neutron spectrometer or a silicon drift detector (used for X-ray fluorescence) tend to have their dimension(s) added to the last dimensions in the data array.

The type of each dimension should should follow the order of scan points, detector pixels, then time-of-flight (i.e. spectroscopy, spectrometry). The rank and dimension sizes (see symbol list) shown here are merely illustrative of coordination between related datasets.

@long_name: (optional) `NX_CHAR`

Title of measurement

@check_sum: (optional) `NX_INT`

Integral of data as check of data integrity

**data_errors**: (optional) `NX_NUMBER` (Rank: 4, Dimensions: [nP, i, j, tof]) {units=`NX_ANY`}


The best estimate of the uncertainty in the data value (array size should match the data field). Where possible, this should be the standard deviation, which has the same units as the data. The form data_error is deprecated.

**x_pixel_offset**: (optional) `NX_FLOAT` (Rank: 2, Dimensions: [i, j]) \{units=`NX_LENGTH`\}
Offset from the detector center in x-direction. Can be multidimensional when needed.

  @axis: (optional) `NX_POSINT`
  Obligatory value: 1

  @primary: (optional) `NX_POSINT`
  Obligatory value: 1

  @long_name: (optional) `NX_CHAR`
  x-axis offset from detector center

**y_pixel_offset**: (optional) `NX_FLOAT` (Rank: 2, Dimensions: [i, j]) \{units=`NX_LENGTH`\}
Offset from the detector center in the y-direction. Can be multidimensional when different values are required for each pixel.

  @axis: (optional) `NX_POSINT`
  Obligatory value: 2

  @primary: (optional) `NX_POSINT`
  Obligatory value: 1

  @long_name: (optional) `NX_CHAR`
  y-axis offset from detector center

**z_pixel_offset**: (optional) `NX_FLOAT` (Rank: 2, Dimensions: [i, j]) \{units=`NX_LENGTH`\}
Offset from the detector center in the z-direction. Can be multidimensional when different values are required for each pixel.

  @axis: (optional) `NX_POSINT`
  Obligatory value: 3

  @primary: (optional) `NX_POSINT`
  Obligatory value: 1

  @long_name: (optional) `NX_CHAR`
  z-axis offset from detector center

**distance**: (optional) `NX_FLOAT` (Rank: 3, Dimensions: [nP, i, j]) \{units=`NX_LENGTH`\}
This is the distance to the previous component in the instrument; most often the sample. The usage depends on the nature of the detector: Most often it is the distance of the detector assembly. But there are irregular detectors. In this case the distance must be specified for each detector pixel.

  Note, it is recommended to use NXtransformations instead.

**polar_angle**: (optional) `NX_FLOAT` (Rank: 3, Dimensions: [nP, i, j]) \{units=`NX_ANGLE`\}
This is the polar angle of the detector towards the previous component in the instrument; most often the sample. The usage depends on the nature of the detector. Most often it is the polar_angle of the detector assembly. But there are irregular detectors. In this case, the polar_angle must be specified for each detector pixel.

Note, it is recommended to use NXtransformations instead.

**azimuthal_angle**: (optional) `NX_FLOAT` (Rank: 3, Dimensions: [nP, i, j]) {units=NX_ANGLE}

This is the azimuthal angle angle of the detector towards the previous component in the instrument; most often the sample. The usage depends on the nature of the detector. Most often it is the azimuthal_angle of the detector assembly. But there are irregular detectors. In this case, the azimuthal_angle must be specified for each detector pixel.

Note, it is recommended to use NXtransformations instead.

**description**: (optional) `NX_CHAR`

name/manufacturer/model/etc. information

**serial_number**: (optional) `NX_CHAR`

Serial number for the detector

**local_name**: (optional) `NX_CHAR`

Local name for the detector

**solid_angle**: (optional) `NX_FLOAT` (Rank: 2, Dimensions: [i, j]) {units=NX_SOLID_ANGLE}

Solid angle subtended by the detector at the sample

**x_pixel_size**: (optional) `NX_FLOAT` (Rank: 2, Dimensions: [i, j]) {units=NX_LENGTH}

Size of each detector pixel. If it is scalar all pixels are the same size.

**y_pixel_size**: (optional) `NX_FLOAT` (Rank: 2, Dimensions: [i, j]) {units=NX_LENGTH}

Size of each detector pixel. If it is scalar all pixels are the same size

**dead_time**: (optional) `NX_FLOAT` (Rank: 3, Dimensions: [nP, i, j]) {units=NX_TIME}

Detector dead time

**gas_pressure**: (optional) `NX_FLOAT` (Rank: 2, Dimensions: [i, j]) {units=NX_PRESSURE}

Detector gas pressure

**detection_gas_path**: (optional) `NX_FLOAT` {units=NX_LENGTH}

maximum drift space dimension

**crate**: (optional) `NX_INT` (Rank: 2, Dimensions: [i, j])

Crate number of detector

@**local_name**: (optional) `NX_CHAR`

Equivalent local term

**slot**: (optional) `NX_INT` (Rank: 2, Dimensions: [i, j])

Slot number of detector

@**local_name**: (optional) `NX_CHAR`

Equivalent local term

**input**: (optional) `NX_INT` (Rank: 2, Dimensions: [i, j])
Input number of detector

@local_name: (optional) NX_CHAR

Equivalent local term

type: (optional) NX_CHAR

Description of type such as He3 gas cylinder, He3 PSD, scintillator, fission chamber, proportion counter, ion chamber, ccd, pixel, image plate, CMOS, ...

real_time: (optional) NX_NUMBER (Rank: 3, Dimensions: [nP, i, j]) {units=NX_TIME}

Real-time of the exposure (use this if exposure time varies for each array element, otherwise use count_time field).

Most often there is a single real time value that is constant across an entire image frame. In such cases, only a 1-D array is needed. But there are detectors in which the real time changes per pixel. In that case, more than one dimension is needed. Therefore the rank of this field should be less than or equal to (detector rank + 1).

start_time: (optional) NX_FLOAT (Rank: 1, Dimensions: [nP]) {units=NX_TIME}

start time for each frame, with the start attribute as absolute reference

@start: (optional) NX_DATE_TIME

stop_time: (optional) NX_FLOAT (Rank: 1, Dimensions: [nP]) {units=NX_TIME}

stop time for each frame, with the start attribute as absolute reference

@start: (optional) NX_DATE_TIME

calibration_date: (optional) NX_DATE_TIME

Date of last calibration (geometry and/or efficiency) measurements

layout: (optional) NX_CHAR

How the detector is represented

Any of these values: point | linear | area

count_time: (optional) NX_NUMBER (Rank: 1, Dimensions: [nP]) {units=NX_TIME}

Elapsed actual counting time

sequence_number: (optional) NX_INT (Rank: 1, Dimensions: [nP])

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.

beam_center_x: (optional) NX_FLOAT {units=NX_LENGTH}

This is the x position where the direct beam would hit the detector. This is a length and can be outside of the actual detector. The length can be in physical units or pixels as documented by the units attribute.

beam_center_y: (optional) NX_FLOAT {units=NX_LENGTH}

This is the y position where the direct beam would hit the detector. This is a length and can be outside of the actual detector. The length can be in physical units or pixels as documented by the units attribute.

frame_start_number: (optional) NX_INT
This is the start number of the first frame of a scan. In protein crystallography measurements one often scans a couple of frames on a give sample, then does something else, then returns to the same sample and scans some more frames. Each time with a new data file. This number helps concatenating such measurements.

**diameter**: (optional) \(NX_FLOAT\) \{units=NX_LENGTH\}

The diameter of a cylindrical detector

**acquisition_mode**: (optional) \(NX_CHAR\)

The acquisition mode of the detector.

Any of these values:

- gated
- triggered
- summed
- event
- histogrammed
- decimated

**angular_calibration_applied**: (optional) \(NX_BOOLEAN\)

True when the angular calibration has been applied in the electronics, false otherwise.

**angular_calibration**: (optional) \(NX_FLOAT\) (Rank: 2, Dimensions: [i, j])

Angular calibration data.

**flatfield_applied**: (optional) \(NX_BOOLEAN\)

True when the flat field correction has been applied in the electronics, false otherwise.

**flatfield**: (optional) \(NX_FLOAT\) (Rank: 2, Dimensions: [i, j])

Flat field correction data.

**flatfield_errors**: (optional) \(NX_FLOAT\) (Rank: 2, Dimensions: [i, j])

Errors of the flat field correction data. The form flatfield_error is deprecated.

**pixel_mask_applied**: (optional) \(NX_BOOLEAN\)

True when the pixel mask correction has been applied in the electronics, false otherwise.

**pixel_mask**: (optional) \(NX_INT\) (Rank: 2, Dimensions: [i, j])

The 32-bit pixel mask for the detector. Can be either one mask for the whole dataset (i.e. an array with indices i, j) or each frame can have its own mask (in which case it would be an array with indices np, i, j).

Contains a bit field for each pixel to signal dead, blind or high or otherwise unwanted or undesirable pixels. They have the following meaning:

- bit 0: gap (pixel with no sensor)
- bit 1: dead
- bit 2: under responding
- bit 3: over responding
- bit 4: noisy
• bit 5: -undefined-
• bit 6: pixel is part of a cluster of problematic pixels (bit set in addition to others)
• bit 7: -undefined-
• bit 8: user defined mask (e.g. around beamstop)
• bits 9-30: -undefined-
• bit 31: virtual pixel (corner pixel with interpolated value)

Normal data analysis software would not take pixels into account when a bit in (mask & 0x0000FFFF) is set. Tag bit in the upper two bytes would indicate special pixel properties that normally would not be a sole reason to reject the intensity value (unless lower bits are set).

If the full bit depths is not required, providing a mask with fewer bits is permissible.

If needed, additional pixel masks can be specified by including additional entries named pixel_mask_N, where N is an integer. For example, a general bad pixel mask could be specified in pixel_mask that indicates noisy and dead pixels, and an additional pixel mask from experiment-specific shadowing could be specified in pixel_mask_2. The cumulative mask is the bitwise OR of pixel_mask and any pixel_mask_N entries.

**image_key**: (optional) `NX_INT` (Rank: 1, Dimensions: [np])

This field allows to distinguish different types of exposure to the same detector “data” field. Some techniques require frequent (re-)calibration inbetween measurements and this way of recording the different measurements preserves the chronological order with is important for correct processing.

This is used for example in tomography (:ref:`NXtomo`) sample projections, dark and flat images, a magic number is recorded per frame.

The key is as follows:
• projection (sample) = 0
• flat field = 1
• dark field = 2
• invalid = 3
• background (no sample, but buffer where applicable) = 4

In cases where the data is of type `NXlog` this can also be an NXlog.

**countrate_correction_applied**: (optional) `NX_BOOLEAN`

Counting detectors usually are not able to measure all incoming particles, especially at higher count-rates. Count-rate correction is applied to account for these errors.

True when count-rate correction has been applied, false otherwise.

**countrate_correction_lookup_table**: (optional) `NX_NUMBER` (Rank: 1, Dimensions: [m])

The `countrate_correction_lookup_table` defines the LUT used for count-rate correction. It maps a measured count \(c\) to its corrected value `countrate_correction_lookup_table[c]`.

\(m\) denotes the length of the table.

**virtual_pixel_interpolation_applied**: (optional) `NX_BOOLEAN`

True when virtual pixel interpolation has been applied, false otherwise.
When virtual pixel interpolation is applied, values of some pixels may contain interpolated values. For example, to account for space between readout chips on a module, physical pixels on edges and corners between chips may have larger sensor areas and counts may be distributed between their logical pixels.

**bit_depth_readout**: (optional) *NX_INT*

How many bits the electronics reads per pixel. With CCD’s and single photon counting detectors, this must not align with traditional integer sizes. This can be 4, 8, 12, 14, 16, …

**detector_readout_time**: (optional) *NX_FLOAT* {units=*NX_TIME*}

Time it takes to read the detector (typically milliseconds). This is important to know for time resolved experiments.

**trigger_delay_time**: (optional) *NX_FLOAT* {units=*NX_TIME*}

Time it takes to start exposure after a trigger signal has been received. This is the reaction time of the detector firmware after receiving the trigger signal to when the detector starts to acquire the exposure, including any user set delay. This is important to know for time resolved experiments.

**trigger_delay_time_set**: (optional) *NX_FLOAT* {units=*NX_TIME*}

User-specified trigger delay.

**trigger_internal_delay_time**: (optional) *NX_FLOAT* {units=*NX_TIME*}

Time it takes to start exposure after a trigger signal has been received. This is the reaction time of the detector hardware after receiving the trigger signal to when the detector starts to acquire the exposure. It forms the lower boundary of the trigger_delay_time when the user does not request an additional delay.

**trigger_dead_time**: (optional) *NX_FLOAT* {units=*NX_TIME*}

Time during which no new trigger signal can be accepted. Typically this is the trigger_delay_time + exposure_time + readout_time. This is important to know for time resolved experiments.

**frame_time**: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_TIME*}

This is time for each frame. This is exposure_time + readout time.

**gain_setting**: (optional) *NX_CHAR*

The gain setting of the detector. This is a detector-specific value meant to document the gain setting of the detector during data collection, for detectors with multiple available gain settings.

Examples of gain settings include:

- standard
- fast
- auto
- high
- medium
- low
- mixed high to medium
- mixed medium to low
Developers are encouraged to use one of these terms, or to submit additional terms to add to the list.

**saturation_value**: (optional) *NX_NUMBER*

The value at which the detector goes into saturation. Especially common to CCD detectors, the data is known to be invalid above this value.

For example, given a saturation_value and an underload_value, the valid pixels are those less than or equal to the saturation_value and greater than or equal to the underload_value.

The precise type should match the type of the data.

**underload_value**: (optional) *NX_NUMBER*

The lowest value at which pixels for this detector would be reasonably measured. The data is known to be invalid below this value.

For example, given a saturation_value and an underload_value, the valid pixels are those less than or equal to the saturation_value and greater than or equal to the underload_value.

The precise type should match the type of the data.

**number_of_cycles**: (optional) *NX_INT*

CCD images are sometimes constructed by summing together multiple short exposures in the electronics. This reduces background etc. This is the number of short exposures used to sum images for an image.

**sensor_material**: (optional) *NX_CHAR*

At times, radiation is not directly sensed by the detector. Rather, the detector might sense the output from some converter like a scintillator. This is the name of this converter material.

**sensor_thickness**: (optional) *NX_FLOAT* `{units=NX_LENGTH}`

At times, radiation is not directly sensed by the detector. Rather, the detector might sense the output from some converter like a scintillator. This is the thickness of this converter material.

**threshold_energy**: (optional) *NX_FLOAT* `{units=NX_ENERGY}`

Single photon counter detectors can be adjusted for a certain energy range in which they work optimally. This is the energy setting for this.

**depends_on**: (optional) *NX_CHAR*

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

The reference point of the detector is the center of the first pixel. In complex geometries the NXoffGeometry groups can be used to provide an unambiguous reference.

**GEOMETRY**: (optional) *NXgeometry*

**DEPRECATED**: Use the field depends_on and NXtransformations to position the detector and NXoffGeometry to describe its shape instead.

Position and orientation of detector

**efficiency**: (optional) *NXdata*
Spectral efficiency of detector with respect to e.g. wavelength

@signal: (optional) `NX_CHAR`
     Obligatory value: `efficiency`

@axes: (optional) `NX_CHAR`
     Any of these values: `. | . | . | . | . | . | . | wavelength`

@wavelength_indices: (optional) `NX_CHAR`
     Obligatory value: `0`

efficiency: (optional) `NX_FLOAT` (Rank: 2, Dimensions: [i, j])
{units=`NX_DIMENSIONLESS`}
     efficiency of the detector

wavelength: (optional) `NX_FLOAT` (Rank: 2, Dimensions: [i, j])
{units=`NX_WAVELENGTH`}
     This field can be two things:
     1. For a pixel detector it provides the nominal wavelength for which the detector has been calibrated.
     2. For other detectors this field has to be seen together with the efficiency field above. For some detectors, the efficiency is wavelength dependent. Thus this field provides the wavelength axis for the efficiency field. In this use case, the efficiency and wavelength arrays must have the same dimensionality.

calibration_method: (optional) `NXnote`
     summary of conversion of array data to pixels (e.g. polynomial approximations) and location of details of the calibrations

data_file: (optional) `NXnote`

COLLECTION: (optional) `NXcollection`
     Use this group to provide other data related to this NXdetector group.

DETECTOR_MODULE: (optional) `NXdetector_module`
     For use in special cases where the data in NXdetector is represented in several parts, each with a separate geometry.

TRANSFORMATIONS: (optional) `NXtransformations`
     This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXdetector/acquisition_mode-field
- /NXdetector/angular_calibration-field
- /NXdetector/angular_calibration_applied-field
- /NXdetector/azimuthal_angle-field
- /NXdetector/beam_center_x-field
- /NXdetector/beam_center_y-field
- /NXdetector/bit_depth_readout-field
- /NXdetector/calibration_date-field
- /NXdetector/calibration_method-group
- /NXdetector/COLLECTION-group
- /NXdetector/count_time-field
- /NXdetector/countrate_correction_applied-field
- /NXdetector/countrate_correction_lookup_table-field
- /NXdetector/crate-field
- /NXdetector/crate@local_name-attribute
- /NXdetector/data-field
- /NXdetector/data@check_sum-attribute
- /NXdetector/data@long_name-attribute
- /NXdetector/data_errors-field
- /NXdetector/data_file-group
- /NXdetector/dead_time-field
- /NXdetector/depends_on-field
- /NXdetector/description-field
- /NXdetector/detection_gas_path-field
- /NXdetector/DETECTOR_MODULE-group
- /NXdetector/detector_number-field
- /NXdetector/detector_readout_time-field
- /NXdetector/diameter-field
- /NXdetector/distance-field
- /NXdetector/efficiency-group
- /NXdetector/efficiency/efficiency-field
- /NXdetector/efficiency/wavelength-field
- /NXdetector/efficiency@axes-attribute
- /NXdetector/efficiency@signal-attribute
• /NXdetector/efficiency@wavelength_indices-attribute
• /NXdetector/flatfield-field
• /NXdetector/flatfield_applied-field
• /NXdetector/flatfield_errors-field
• /NXdetector/frame_start_number-field
• /NXdetector/frame_time-field
• /NXdetector/gain_setting-field
• /NXdetector/gas_pressure-field
• /NXdetector/GEOMETRY-group
• /NXdetector/image_key-field
• /NXdetector/input-field
• /NXdetector/input@local_name-attribute
• /NXdetector/layout-field
• /NXdetector/local_name-field
• /NXdetector/number_of_cycles-field
• /NXdetector/pixel_mask-field
• /NXdetector/pixel_mask_applied-field
• /NXdetector/polar_angle-field
• /NXdetector/raw_time_of_flight-field
• /NXdetector/raw_time_of_flight@frequency-attribute
• /NXdetector/real_time-field
• /NXdetector/saturation_value-field
• /NXdetector/sensor_material-field
• /NXdetector/sensor_thickness-field
• /NXdetector/sequence_number-field
• /NXdetector/serial_number-field
• /NXdetector/slot-field
• /NXdetector/slot@local_name-attribute
• /NXdetector/solid_angle-field
• /NXdetector/start_time-field
• /NXdetector/start_time@start-attribute
• /NXdetector/stop_time-field
• /NXdetector/stop_time@start-attribute
• /NXdetector/threshold_energy-field
• /NXdetector/time_of_flight-field
• /NXdetector/time_of_flight@axis-attribute
• /NXdetector/time_of_flight@long_name-attribute
• /NXdetector/time_of_flight@primary-attribute
• /NXdetector/TRANSFORMATIONS-group
• /NXdetector/trigger_dead_time-field
• /NXdetector/trigger_delay_time-field
• /NXdetector/trigger_delay_time_set-field
• /NXdetector/trigger_internal_delay_time-field
• /NXdetector/type-field
• /NXdetector/underload_value-field
• /NXdetector/virtual_pixel_interpolation_applied-field
• /NXdetector/x_pixel_offset-field
• /NXdetector/x_pixel_offset@axis-attribute
• /NXdetector/x_pixel_offset@long_name-attribute
• /NXdetector/x_pixel_offset@primary-attribute
• /NXdetector/x_pixel_size-field
• /NXdetector/y_pixel_offset-field
• /NXdetector/y_pixel_offset@axis-attribute
• /NXdetector/y_pixel_offset@long_name-attribute
• /NXdetector/y_pixel_offset@primary-attribute
• /NXdetector/y_pixel_size-field
• /NXdetector/z_pixel_offset-field
• /NXdetector/z_pixel_offset@axis-attribute
• /NXdetector/z_pixel_offset@long_name-attribute
• /NXdetector/z_pixel_offset@primary-attribute
• /NXdetector@default-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXdetector.nxdl.xml

NXdetector_group

Status:
base class, extends NXobject

Description:
Logical grouping of detectors. When used, describes a group of detectors.

Each detector is represented as an NXdetector with its own detector data array. Each detector data array may be further decomposed into array sections by use of NXdetector_module groups. Detectors can be grouped logically together using NXdetector_group. Groups can be further grouped hierarchically in a single NXdetector_group (for example, if there are multiple detectors at an endstation or multiple endstations at a facility). Alternatively, multiple NXdetector_groups can be provided.
The groups are defined hierarchically, with names given in the group_names field, unique identifying indices given in the field group_index, and the level in the hierarchy given in the group_parent field. For example if an x-ray detector group, DET, consists of four detectors in a rectangular array:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DTL</td>
<td>DTR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLL</td>
<td>DLR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We could have:

```
group_names: ["DET", "DTL", "DTR", "DLL", "DLR"]
group_index: [1, 2, 3, 4, 5]
group_parent: [-1, 1, 1, 1, 1]
```

Symbols:

No symbol table

Groups cited:

none

Structure:

@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

group_names: (optional) NX_CHAR

An array of the names of the detectors given in NXdetector groups or the names of hierarchical groupings of detectors given as names of NXdetector_group groups or in NXdetector_group group_names and group_parent fields as having children.

group_index: (optional) NX_INT (Rank: 1, Dimensions: [i])

An array of unique identifiers for detectors or groupings of detectors.

Each ID is a unique ID for the corresponding detector or group named in the field group_names.

The IDs are positive integers starting with 1.

group_parent: (optional) NX_INT (Rank: same as field group_index, Dimensions: same as field group_index)

An array of the hierarchical levels of the parents of detectors or groupings of detectors.

A top-level grouping has parent level -1.

group_type: (optional) NX_INT (Rank: same as field group_index, Dimensions: same as field group_index)

Code number for group type, e.g. bank=1, tube=2 etc.
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXdetector_group/group_index-field
- /NXdetector_group/group_names-field
- /NXdetector_group/group_parent-field
- /NXdetector_group/group_type-field
- /NXdetector_group@default-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXdetector_group.nxdl.xml

NXdetector_module

Status:

base class, extends NXobject

Description:

Geometry and logical description of a detector module. When used, child group to NXdetector. Many detectors consist of multiple smaller modules. Sometimes it is important to know the exact position of such modules. This is the purpose of this group. It is a child group to NXdetector.

Note, the pixel size is given as values in the array fast_pixel_direction and slow_pixel_direction.

Symbols:

No symbol table

Groups cited:

none

Structure:

@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

data_origin: (optional) NX_INT

A dimension-2 or dimension-3 field which gives the indices of the origin of the hyperslab of data for this module in the main area detector image in the parent NXdetector module.

The data_origin is 0-based.

The frame number dimension (np) is omitted. Thus the data_origin field for a dimension-2 dataset with indices (np, i, j) will be an array with indices (i, j), and for a dimension-3 dataset with indices (np, i, j, k) will be an array with indices (i, j, k).

The order of indices (i, j or i, j, k) is slow to fast.

data_size: (optional) NX_INT
Two or three values for the size of the module in pixels in each direction. Dimensionality and order of indices is the same as for data_origin.

**module_offset**: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Offset of the module in regards to the origin of the detector in an arbitrary direction.

  - @transformation_type: (optional) *NX_CHAR*
    - Obligatory value: translation
  - @vector: (optional) *NX_NUMBER*
    - Three values that define the axis for this transformation
  - @offset: (optional) *NX_NUMBER*
    - A fixed offset applied before the transformation (three vector components).
  - @offset_units: (optional) *NX_CHAR*
    - Units of the offset.
  - @depends_on: (optional) *NX_CHAR*
    - Points to the path of the next element in the geometry chain.

**fast_pixel_direction**: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Values along the direction of *fastest varying* pixel direction. Each value in this array is the size of a pixel in the units specified. Alternatively, if only one value is given, all pixels in this direction have the same value. The direction itself is given through the vector attribute.

  - @transformation_type: (optional) *NX_CHAR*
    - Obligatory value: translation
  - @vector: (optional) *NX_NUMBER*
    - Three values that define the axis for this transformation
  - @offset: (optional) *NX_NUMBER*
    - A fixed offset applied before the transformation (three vector components).
  - @offset_units: (optional) *NX_CHAR*
    - Units of the offset.
  - @depends_on: (optional) *NX_CHAR*
    - Points to the path of the next element in the geometry chain.

**slow_pixel_direction**: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Values along the direction of *slowest varying* pixel direction. Each value in this array is the size of a pixel in the units specified. Alternatively, if only one value is given, all pixels in this direction have the same value. The direction itself is given through the vector attribute.

  - @transformation_type: (optional) *NX_CHAR*
    - Obligatory value: translation
  - @vector: (optional) *NX_NUMBER*
    - Three values that define the axis for this transformation
  - @offset: (optional) *NX_NUMBER*
    - A fixed offset applied before the transformation (three vector components).
@offset_units: (optional) **NX_CHAR**

Units of the offset.

@depends_on: (optional) **NX_CHAR**

Points to the path of the next element in the geometry chain.

depends_on: (optional) **NX_CHAR**

Points to the start of the dependency chain for this module.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXdetector_module/data_origin-field`
- `/NXdetector_module/data_size-field`
- `/NXdetector_module/depends_on-field`
- `/NXdetector_module/fast_pixel_direction-field`
- `/NXdetector_module/fast_pixel_direction@depends_on-attribute`
- `/NXdetector_module/fast_pixel_direction@offset-attribute`
- `/NXdetector_module/fast_pixel_direction@offset_units-attribute`
- `/NXdetector_module/fast_pixel_direction@transformation_type-attribute`
- `/NXdetector_module/fast_pixel_direction@vector-attribute`
- `/NXdetector_module/module_offset-field`
- `/NXdetector_module/module_offset@depends_on-attribute`
- `/NXdetector_module/module_offset@offset-attribute`
- `/NXdetector_module/module_offset@offset_units-attribute`
- `/NXdetector_module/module_offset@transformation_type-attribute`
- `/NXdetector_module/module_offset@vector-attribute`
- `/NXdetector_module/slow_pixel_direction-field`
- `/NXdetector_module/slow_pixel_direction@depends_on-attribute`
- `/NXdetector_module/slow_pixel_direction@offset-attribute`
- `/NXdetector_module/slow_pixel_direction@offset_units-attribute`
- `/NXdetector_module/slow_pixel_direction@transformation_type-attribute`
- `/NXdetector_module/slow_pixel_direction@vector-attribute`
- `/NXdetector_module@default-attribute`

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/base_classes/NXdetector_module.nxdl.xml
**NXdisk_chopper**

**Status:**
base class, extends *NXobject*

**Description:**
A device blocking the beam in a temporal periodic pattern.
A disk which blocks the beam but has one or more slits to periodically let neutrons through as the disk rotates. Often used in pairs, one NXdisk_chopper should be defined for each disk.
The rotation of the disk is commonly monitored by recording a timestamp for each full rotation of disk, by having a sensor in the stationary disk housing sensing when it is aligned with a feature (such as a magnet) on the disk. We refer to this below as the “top-dead-center signal”.
Angles and positive rotation speeds are measured in an anticlockwise direction when facing away from the source.

**Symbols:**
This symbol will be used below to coordinate datasets with the same shape.

\[ n \]  Number of slits in the disk

**Groups cited:**
NXgeometry, NXoff_geometry, NXtransformations

**Structure:**

@default: (optional) *NX_CHAR*
Declares which child group contains a path leading to a *NXdata* group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See [https://www.nexusformat.org/2014_How_to_find_default_data.html](https://www.nexusformat.org/2014_How_to_find_default_data.html) for a summary of the discussion.

\[ type \]: (optional) *NX_CHAR*
Type of the disk-chopper: only one from the enumerated list (match text exactly)
Any of these values:
• Chopper type single
• contra_rotating_pair
• synchro_pair
rotation_speed: (optional) *NX_FLOAT* \{{units=*NX_FREQUENCY*}\}
Chopper rotation speed. Positive for anticlockwise rotation when facing away from the source, negative otherwise.

\[ slits \]: (optional) *NX_INT*
Number of slits

\[ slit_angle \]: (optional) *NX_FLOAT* \{{units=*NX_ANGLE*}\}
Angular opening

\[ pair_separation \]: (optional) *NX_FLOAT* \{{units=*NX_LENGTH*}\}
Disk spacing in direction of beam
slit_edges: (optional) NX_FLOAT (Rank: 1, Dimensions: [2n]) {units=NX_ANGLE}

Angle of each edge of every slit from the position of the top-dead-center timestamp sensor, anticlockwise when facing away from the source. The first edge must be the opening edge of a slit, thus the last edge may have an angle greater than 360 degrees.

top_dead_center: (optional) NX_NUMBER {units=NX_TIME}

Timestamps of the top-dead-center signal. The times are relative to the “start” attribute and in the units specified in the “units” attribute. Please note that absolute timestamps under unix are relative to 1970-01-01T00:00:00.0Z.

@start: (optional) NX_DATE_TIME

beam_position: (optional) NX_FLOAT {units=NX_ANGLE}

Angular separation of the center of the beam and the top-dead-center timestamp sensor, anticlockwise when facing away from the source.

radius: (optional) NX_FLOAT {units=NX_LENGTH}

Radius of the disk

slit_height: (optional) NX_FLOAT {units=NX_LENGTH}

Total slit height

phase: (optional) NX_FLOAT {units=NX_ANGLE}

Chopper phase angle

delay: (optional) NX_NUMBER {units=NX_TIME}

Time difference between timing system t0 and chopper driving clock signal

ratio: (optional) NX_INT

Pulse reduction factor of this chopper in relation to other choppers/fastest pulse in the instrument

distance: (optional) NX_FLOAT {units=NX_LENGTH}

Effective distance to the origin. Note, it is recommended to use NXtransformations instead.

wavelength_range: (optional) NX_FLOAT (Rank: 1, Dimensions: [2]) {units=NX_WAVELENGTH}

Low and high values of wavelength range transmitted

depends_on: (optional) NX_CHAR

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

The reference plane of the disk chopper includes the surface of the spinning disk which faces the source. The reference point in the x and y axis is the point on this surface which is the centre of the axle which the disk is spinning around. The reference plane is orthogonal to the z axis and its position is the reference point on that axis.

Note: This reference point in almost all practical cases is not where the beam passes though.

3.3. NeXus Class Definitions
GEOMETRY: (optional) NXgeometry

DEPRECATED: Use the field depends_on and NXtransformations to position the chopper and NXoff_geometry to describe its shape instead

OFF_GEOMETRY: (optional) NXoff_geometry

This group describes the shape of the beam line component

TRANSFORMATIONS: (optional) NXtransformations

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXdisk_chopper/beam_position-field
- /NXdisk_chopper/delay-field
- /NXdisk_chopper/depends_on-field
- /NXdisk_chopper/distance-field
- /NXdisk_chopper/GEOMETRY-group
- /NXdisk_chopper/OFF_GEOMETRY-group
- /NXdisk_chopper/pair_separation-field
- /NXdisk_chopper/phase-field
- /NXdisk_chopper/radius-field
- /NXdisk_chopper/ratio-field
- /NXdisk_chopper/rotation_speed-field
- /NXdisk_chopper/slit_angle-field
- /NXdisk_chopper/slit_edges-field
- /NXdisk_chopper/slit_height-field
- /NXdisk_chopper/slits-field
- /NXdisk_chopper/top_dead_center-field
- /NXdisk_chopper/top_dead_center@start-attribute
- /NXdisk_chopper/TRANSFORMATIONS-group
- /NXdisk_chopper/type-field
- /NXdisk_chopper/wavelength_range-field
- /NXdisk_chopper@default-attribute
NXentry

Status:
base class, extends NXobject

Description:
(required) NXentry describes the measurement.
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.

Symbols:
No symbol table

Groups cited:
NXcollection, NXdata, NXinstrument, NXmonitor, NXnote, NXparameters, NXprocess, NXsample, NXsubentry, NXuser

Structure:
@default: (optional) NX_CHAR
Declares which NXdata group contains the data to be shown by default. It is used to resolve ambiguity when one NXdata group exists. The value names a child group. If that group itself has a default attribute, continue this chain until an NXdata group is reached.

For more information about how NeXus identifies the default plottable data, see the Find Plottable Data, v3 section.

@IDF_Version: (optional) NX_CHAR
ISIS Muon IDF_Version

title: (optional) NX_CHAR
Extended title for entry

experiment_identifier: (optional) NX_CHAR
Unique identifier for the experiment, defined by the facility, possibly linked to the proposals

experiment_description: (optional) NX_CHAR
Brief summary of the experiment, including key objectives.

collection_identifier: (optional) NX_CHAR
User or Data Acquisition defined group of NeXus files or NXentry

collection_description: (optional) NX_CHAR
Brief summary of the collection, including grouping criteria.

title: (optional) NX_CHAR
unique identifier for the measurement, defined by the facility.

title: (optional) NX_CHAR
UUID identifier for the measurement.

@version: (optional) `NX_CHAR`

Version of UUID used

features: (optional) `NX_CHAR`

Reserved for future use by NIAC.

See https://github.com/nexusformat/definitions/issues/382

definition: (optional) `NX_CHAR`

(alternate use: see same field in `NXsubentry` for preferred)

Official NeXus NXDL schema to which this entry conforms which must be the name of the NXDL file (case sensitive without the file extension) that the NXDL schema is defined in.

For example the definition field for a file that conformed to the `NXarpes.nxdl.xml` definition must contain the string `NXarpes`.

This field is provided so that `NXentry` can be the overlay position in a NeXus data file for an application definition and its set of groups, fields, and attributes.

It is advised to use `NXsubentry`, instead, as the overlay position.

@version: (optional) `NX_CHAR`

NXDL version number

@URL: (optional) `NX_CHAR`

URL of NXDL file

definition_local: (optional) `NX_CHAR`

DEPRECATED: see same field in `NXsubentry` for preferred use

Local NXDL schema extended from the entry specified in the definition field. This contains any locally-defined, additional fields in the entry.

@version: (optional) `NX_CHAR`

NXDL version number

@URL: (optional) `NX_CHAR`

URL of NXDL file

start_time: (optional) `NX_DATE_TIME`

Starting time of measurement

duration: (optional) `NX_INT` `{units=NX_TIME}`

Duration of measurement

collection_time: (optional) `NX_FLOAT` `{units=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: (optional) `NX_CHAR`

Such as “2007-3”. Some user facilities organize their beam time into run cycles.
program_name: (optional) NX_CHAR
   Name of program used to generate this file
@version: (optional) NX_CHAR
   Program version number
@configuration: (optional) NX_CHAR
   configuration of the program
revision: (optional) NX_CHAR
   Revision id of the file due to re-calibration, reprocessing, new analysis, new instrument defini-
   tion format, . . .
@comment: (optional) NX_CHAR
pre_sample_flightpath: (optional) NX_FLOAT [units=NX_LENGTH]
   This is the flightpath before the sample position. This can be determined by a chopper, by the
   moderator or the source itself. In other words: it the distance to the component which gives
   the T0 signal to the detector electronics. If another component in the NXinstrument hierarchy
   provides this information, this should be a link.
DATA: (optional) NXdata
   The data group

Note: Before the NIAC2016 meeting\(^1\), at least one NXdata group was required in each NXentry
   group. At the NIAC2016 meeting, it was decided to make NXdata an optional group in NXentry
groups for data files that do not use an application definition. It is recommended strongly that
all NeXus data files provide a NXdata group. It is permissable to omit the NXdata group only
when defining the default plot is not practical or possible from the available data.

For example, neutron event data may not have anything that makes a useful plot without exten-
sive processing.

Certain application definitions override this decision and require an NXdata group in the NX-
entry group. The minOccurs=0 attribute in the application definition will indicate the NXdata
   group is optional, otherwise, it is required.

experiment_documentation: (optional) NXnote
   Description of the full experiment (document in pdf, latex, . . .)
notes: (optional) NXnote
   Notes describing entry
thumbnail: (optional) 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.
@type: (optional) NX_CHAR
   The mime type should be an image/*

Obligatory value: image/*

---

USER: (optional) $NXuser$
SAMPLE: (optional) $NXsample$
INSTRUMENT: (optional) $NXinstrument$
COLLECTION: (optional) $NXcollection$
MONITOR: (optional) $NXmonitor$
PARAMETERS: (optional) $NXparameters$
PROCESS: (optional) $NXprocess$
SUBENTRY: (optional) $NXsubentry$

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXentry/COLLECTION-group
- /NXentry/collection_description-field
- /NXentry/collection_identifier-field
- /NXentry/collection_time-field
- /NXentry/DATA-group
- /NXentry/definition-field
- /NXentry/definition@URL-attribute
- /NXentry/definition@version-attribute
- /NXentry/definition_local-field
- /NXentry/definition_local@URL-attribute
- /NXentry/definition_local@version-attribute
- /NXentry/duration-field
- /NXentry/end_time-field
- /NXentry/entry_identifier-field
- /NXentry/entry_identifier_uuid-field
- /NXentry/entry_identifier_uuid@version-attribute
- /NXentry/experiment_description-field
- /NXentry/experiment_documentation-group
- /NXentry/experiment_identifier-field
- /NXentry/features-field
- /NXentry/INSTRUMENT-group
- /NXentry/MONITOR-group
- /NXentry/notes-group
- /NXentry/PARAMETERS-group
- /NXentry/pre_sample_flightpath-field
• /NXentry/PROCESS-group
• /NXentry/program_name-field
• /NXentry/program_name@configuration-attribute
• /NXentry/program_name@version-attribute
• /NXentry/revision-field
• /NXentry/revision@comment-attribute
• /NXentry/run_cycle-field
• /NXentry/SAMPLE-group
• /NXentry/start_time-field
• /NXentry/SUBENTRY-group
• /NXentry/thumbnail-group
• /NXentry/thumbnail@type-attribute
• /NXentry/title-field
• /NXentry/USER-group
• /NXentry@default-attribute
• /NXentry@IDF_Version-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXentry.nxdl.xml

NXenvironment

Status:
base class, extends NXobject

Description:
Parameters for controlling external conditions

Symbols:
No symbol table

Groups cited:
NXgeometry, NXnote, NXsensor, NXtransformations

Structure:

name: (optional) NX_CHAR
Apparatus identification code/model number; e.g. OC100 011

short_name: (optional) NX_CHAR
Alternative short name, perhaps for dashboard display like a present Seblock name

type: (optional) NX_CHAR
Type of apparatus. This could be the SE codes in scheduling database; e.g. OC/100

description: (optional) NX_CHAR
Description of the apparatus; e.g. 100mm bore orange cryostat with Roots pump

3.3. NeXus Class Definitions
**program**: (optional) *NX_CHAR*

Program controlling the apparatus; e.g. LabView VI name

**depends_on**: (optional) *NX_CHAR*

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

**position**: (optional) *NXgeometry*

The position and orientation of the apparatus. Note, it is recommended to use NXtransformations instead.

**TRANSFORMATIONS**: (optional) *NXtransformations*

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

**NOTE**: (optional) *NXnote*

Additional information, LabView logs, digital photographs, etc

**SENSOR**: (optional) *NXsensor*

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXenvironment/depends_on-field
- /NXenvironment/description-field
- /NXenvironment/name-field
- /NXenvironment/NODE-group
- /NXenvironment/position-group
- /NXenvironment/program-field
- /NXenvironment/SENSOR-group
- /NXenvironment/short_name-field
- /NXenvironment/TRANSFORMATIONS-group
- /NXenvironment/type-field

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/base_classes/NXenvironment.nxdl.xml
**NXevent_data**

**Status:**

base class, extends *NXobject*

**Description:**

NXevent_data is a special group for storing data from neutron detectors in event mode. In this mode, the detector electronics emits a stream of detectorID, timestamp pairs. With detectorID describing the detector element in which the neutron was detected and timestamp the timestamp at which the neutron event was detected. In NeXus detectorID maps to event_id, event_time_offset to the timestamp.

As this kind of data is common at pulsed neutron sources, the timestamp is almost always relative to the start of a neutron pulse. Thus the pulse timestamp is recorded too together with an index in the event_id, event_time_offset pair at which data for that pulse starts. At reactor source the same pulsed data effect may be achieved through the use of choppers or in stroboscopic measurement setups.

In order to make random access to timestamped data faster there is an optional array pair of cue_timestamp_zero and cue_index. The cue_timestamp_zero will contain courser timestamps then in the time array, say every five minutes. The cue_index will then contain the index into the event_id, event_time_offset pair of arrays for that courser cue_timestamp_zero.

**Symbols:**

No symbol table

**Groups cited:**

none

**Structure:**

@default: (optional) *NX_CHAR*

Declares which child group contains a path leading to a *NXdata* group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

**event_time_offset:** (optional) *NX_NUMBER* (Rank: 1, Dimensions: [i]) {units=NX_TIME_OF_FLIGHT}

A list of timestamps for each event as it comes in.

**event_id:** (optional) *NX_INT* (Rank: 1, Dimensions: [i]) {units=NX_DIMENSIONLESS}

There will be extra information in the NXdetector to convert event_id to detector_number.

**event_time_zero:** (optional) *NX_NUMBER* (Rank: 1, Dimensions: [j]) {units=NX_TIME}

The time that each pulse started with respect to the offset

@offset: (optional) *NX_DATE_TIME*

ISO8601

**event_index:** (optional) *NX_INT* (Rank: 1, Dimensions: [j]) {units=NX_DIMENSIONLESS}

The index into the event_time_offset, event_id pair for the pulse occurring at the matching entry in event_time_zero.

**pulse_height:** (optional) *NX_FLOAT* (Rank: 2, Dimensions: [i, k]) {units=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.

**cue_timestamp_zero**: (optional) `NX_DATE_TIME` `{units=NX_TIME}`

Timestamps matching the corresponding cue_index into the event_id, event_time_offset pair.

`@start`: (optional) `NX_DATE_TIME`

**cue_index**: (optional) `NX_INT`

Index into the event_id, event_time_offset pair matching the corresponding cue_timestamp.

### Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXevent_data/cue_index-field`
- `/NXevent_data/cue_timestamp_zero-field`
- `/NXevent_data/cue_timestamp_zero@start-attribute`
- `/NXevent_data/event_id-field`
- `/NXevent_data/event_index-field`
- `/NXevent_data/event_time_offset-field`
- `/NXevent_data/event_time_zero-field`
- `/NXevent_data/event_time_zero@offset-attribute`
- `/NXevent_data/pulse_height-field`
- `/NXevent_data@default-attribute`

### NXDL Source:

https://github.com/nexusformat/definitions/blob/main/base_classes/NXevent_data.nxdl.xml

### NXfermi_chopper

**Status:**

base class, extends `NXObject`

**Description:**

A Fermi chopper, possibly with curved slits.

**Symbols:**

No symbol table

**Groups cited:**

`NXgeometry`, `NXoff_geometry`, `NXtransformations`

**Structure:**

`@default`: (optional) `NX_CHAR`
Declares which child group contains a path leading to a `NXdata` group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See [https://www.nexusformat.org/2014_How_to_find_default_data.html](https://www.nexusformat.org/2014_How_to_find_default_data.html) for a summary of the discussion.

type: (optional) `NX_CHAR`

Fermi chopper type

rotation_speed: (optional) `NX_FLOAT` `units=NX_FREQUENCY`

chopper rotation speed

radius: (optional) `NX_FLOAT` `units=NX_LENGTH`

radius of chopper

slit: (optional) `NX_FLOAT` `units=NX_LENGTH`

width of an individual slit

r_slit: (optional) `NX_FLOAT` `units=NX_LENGTH`

radius of curvature of slits

number: (optional) `NX_INT` `units=NX_UNITLESS`

number of slits

height: (optional) `NX_FLOAT` `units=NX_LENGTH`

input beam height

width: (optional) `NX_FLOAT` `units=NX_LENGTH`

input beam width

distance: (optional) `NX_FLOAT` `units=NX_LENGTH`

distance. Note, it is recommended to use NXtransformations instead.

wavelength: (optional) `NX_FLOAT` `units=NX_WAVELENGTH`

Wavelength transmitted by chopper

energy: (optional) `NX_FLOAT` `units=NX_ENERGY`

energy selected

absorbing_material: (optional) `NX_CHAR`

absorbing material

transmitting_material: (optional) `NX_CHAR`

transmitting material

depends_on: (optional) `NX_CHAR`

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

GEOMETRY: (optional) `NXgeometry`
DEPRECATED: Use the field depends_on and NXtransformations to position the chopper and NXoff_geometry to describe its shape instead.

geometry of the fermi chopper

OFF GEOMETRY: (optional) NXoff_geometry

This group describes the shape of the beam line component

TRANSFORMATIONS: (optional) NXtransformations

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXfermi_chopper/absorbing_material-field
- /NXfermi_chopper/depends_on-field
- /NXfermi_chopper/distance-field
- /NXfermi_chopper/energy-field
- /NXfermi_chopper/GEOMETRY-group
- /NXfermi_chopper/height-field
- /NXfermi_chopper/number-field
- /NXfermi_chopper/OFF_GEOMETRY-group
- /NXfermi_chopper/r_slit-field
- /NXfermi_chopper/radius-field
- /NXfermi_chopper/rotation_speed-field
- /NXfermi_chopper/slit-field
- /NXfermi_chopper/TRANSFORMATIONS-group
- /NXfermi_chopper/transmitting_material-field
- /NXfermi_chopper/type-field
- /NXfermi_chopper/wavelength-field
- /NXfermi_chopper/width-field
- /NXfermi_chopper@default-attribute

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/base_classes/NXfermi_chopper.nxdl.xml
NXfilter

Status:

base class, extends NXobject

Description:

For band pass beam filters.

If uncertain whether to use NXfilter (band-pass filter) or NXattenuator (reduces beam intensity), then use NXattenuator.

Symbols:

No symbol table

Groups cited:

NXdata, NXgeometry, NXlog, NXoff_geometry, NXsensor, NXtransformations

Structure:

@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

description: (optional) NX_CHAR

Composition of the filter. Chemical formula can be specified separately.

This field was changed (2010-11-17) from an enumeration to a string since common usage showed a wider variety of use than a simple list. These are the items in the list at the time of the change: Beryllium | Pyrolytic Graphite | Graphite | Sapphire | Silicon | Supermirror.

status: (optional) NX_CHAR

position with respect to in or out of the beam (choice of only “in” or “out”)

Any of these values:

  • in: in the beam
  • out: out of the beam

temperature: (optional) NX_FLOAT {units=NX_TEMPERATURE}

average/nominal filter temperature

thickness: (optional) NX_FLOAT {units=NX_LENGTH}

Thickness of the filter

density: (optional) NX_NUMBER {units=NX_MASS_DENSITY}

mass density of the filter

chemical_formula: (optional) 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 alphabetical order of their symbol.

• This is the Hill system used by Chemical Abstracts.

unit_cell_a: (optional) NX_FLOAT {units=NX_LENGTH}
  Unit cell lattice parameter: length of side a

unit_cell_b: (optional) NX_FLOAT {units=NX_LENGTH}
  Unit cell lattice parameter: length of side b

unit_cell_c: (optional) NX_FLOAT {units=NX_LENGTH}
  Unit cell lattice parameter: length of side c

unit_cell_alpha: (optional) NX_FLOAT {units=NX_ANGLE}
  Unit cell lattice parameter: angle alpha

unit_cell_beta: (optional) NX_FLOAT {units=NX_ANGLE}
  Unit cell lattice parameter: angle beta

unit_cell_gamma: (optional) NX_FLOAT {units=NX_ANGLE}
  Unit cell lattice parameter: angle gamma

unit_cell_volume: (optional) NX_FLOAT (Rank: 1, Dimensions: [n_comp]) {units=NX_VOLUME}
  Unit cell

orientation_matrix: (optional) NX_FLOAT (Rank: 3, Dimensions: [n_comp, 3, 3])

m_value: (optional) NX_FLOAT {units=NX_DIMENSIONLESS}
  m value of supermirror filter

substrate_material: (optional) NX_CHAR
  Substrate material of supermirror filter

substrate_thickness: (optional) NX_FLOAT {units=NX_LENGTH}
  Substrate thickness of supermirror filter

coating_material: (optional) NX_CHAR
  Coating material of supermirror filter

substrate_roughness: (optional) NX_FLOAT {units=NX_LENGTH}
  Substrate roughness (RMS) of supermirror filter
coating_roughness: (optional) NX_FLOAT (Rank: 1, Dimensions: [nsurf]) {units=NX_LENGTH}
coating roughness (RMS) of supermirror filter

depends_on: (optional) NX_CHAR
NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

GEOMETRY: (optional) NXgeometry

DEPRECATED: Use the field depends_on and NXtransformations to filter the beamstop and NXoff_geometry to describe its shape instead

Geometry of the filter

transmission: (optional) NXdata
Wavelength transmission profile of filter

temperature_log: (optional) NXlog
Linked temperature_log for the filter

sensor_type: (optional) NXsensor
Sensor(s) used to monitor the filter temperature

OFF_GEOMETRY: (optional) NXoff_geometry
This group describes the shape of the beam line component

TRANSFORMATIONS: (optional) NXtransformations
This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXfilter/chemical_formula-field
- /NXfilter/coating_material-field
- /NXfilter/coating_roughness-field
- /NXfilter/density-field
- /NXfilter/depends_on-field
- /NXfilter/description-field
- /NXfilter/GEOMETRY-group
- /NXfilter/m_value-field
- /NXfilter/OFF_GEOMETRY-group
- /NXfilter/orientation_matrix-field
- /NXfilter/sensor_type-group
• /NXfilter/status-field
• /NXfilter/substrate_material-field
• /NXfilter/substrate_roughness-field
• /NXfilter/substrate_thickness-field
• /NXfilter/temperature-field
• /NXfilter/temperature_log-group
• /NXfilter/thickness-field
• /NXfilter/TRANSFORMATIONS-group
• /NXfilter/transmission-group
• /NXfilter/unit_cell_a-field
• /NXfilter/unit_cell_alpha-field
• /NXfilter/unit_cell_b-field
• /NXfilter/unit_cell_beta-field
• /NXfilter/unit_cell_c-field
• /NXfilter/unit_cell_gamma-field
• /NXfilter/unit_cell_volume-field
• /NXfilter@default-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXfilter.nxdl.xml

NXflipper

Status:
base class, extends NXobject

Description:
A spin flipper.

Symbols:
No symbol table

Groups cited:
NXtransformations

Structure:
@default: (optional) NX_CHAR
Declares which child group contains a path leading to a NXdata group.
It is recommended (as of NIAC2014) to use this attribute to help define the path to the default
dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html
for a summary of the discussion.

type: (optional) NX_CHAR
Any of these values: coil | current-sheet
flip_turns: (optional) NX_FLOAT {units=NX_PER_LENGTH}
  Linear density of turns (such as number of turns/cm) in flipping field coils

comp_turns: (optional) NX_FLOAT {units=NX_PER_LENGTH}
  Linear density of turns (such as number of turns/cm) in compensating field coils

guide_turns: (optional) NX_FLOAT {units=NX_PER_LENGTH}
  Linear density of turns (such as number of turns/cm) in guide field coils

flip_current: (optional) NX_FLOAT {units=NX_CURRENT}
  Flipping field coil current in “on” state

comp_current: (optional) NX_FLOAT {units=NX_CURRENT}
  Compensating field coil current in “on” state

guide_current: (optional) NX_FLOAT {units=NX_CURRENT}
  Guide field coil current in “on” state

thickness: (optional) NX_FLOAT {units=NX_LENGTH}
  thickness along path of neutron travel

depends_on: (optional) NX_CHAR
  NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

TRANSFORMATIONS: (optional) NXtransformations
  This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXflipper/comp_current-field
- /NXflipper/comp_turns-field
- /NXflipper/depends_on-field
- /NXflipper/flip_current-field
- /NXflipper/flip_turns-field
- /NXflipper/guide_current-field
- /NXflipper/guide_turns-field
- /NXflipper/thickness-field
- /NXflipper/TRANSFORMATIONS-group
- /NXflipper/type-field
- /NXflipper@default-attribute
NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXflipper.nxdl.xml

NXfresnel_zone_plate

Status:
base class, extends NXobject

Description:
A fresnel zone plate

Symbols:
No symbol table

Groups cited:
NXtransformations

Structure:

@default: (optional) NX_CHAR
Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

focus_parameters: (optional) NX_FLOAT (Rank: 1)
list of polynomial coefficients describing the focal length of the zone plate, in increasing powers of photon energy, that describes the focal length of the zone plate (in microns) at an X-ray photon energy (in electron volts).

outer_diameter: (optional) NX_FLOAT {units=NX_LENGTH}

outermost_zone_width: (optional) NX_FLOAT {units=NX_LENGTH}

central_stop_diameter: (optional) NX_FLOAT {units=NX_LENGTH}

fabrication: (optional) NX_CHAR
how the zone plate was manufactured
Any of these values: etched | plated | zone doubled | other

zone_height: (optional) NX_FLOAT {units=NX_LENGTH}

zone_material: (optional) NX_CHAR
Material of the zones themselves

zone_support_material: (optional) NX_CHAR
Material present between the zones. This is usually only present for the “zone doubled” fabrication process

central_stop_material: (optional) NX_CHAR

central_stop_thickness: (optional) NX_FLOAT {units=NX_LENGTH}

mask_thickness: (optional) NX_FLOAT {units=NX_LENGTH}

mask_material: (optional) NX_CHAR
If no mask is present, set mask_thickness to 0 and omit the mask_material field

**support_membrane_material**: (optional) *NX_CHAR*

**support_membrane_thickness**: (optional) *NX_FLOAT* \{ units=NX_LENGTH \}

**depends_on**: (optional) *NX_CHAR*

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “." if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

**TRANSFORMATIONS**: (optional) *NXtransformations*

“Engineering” position of the fresnel zone plate

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXfresnel_zone_plate/central_stop_diameter-field`
- `/NXfresnel_zone_plate/central_stop_material-field`
- `/NXfresnel_zone_plate/central_stop_thickness-field`
- `/NXfresnel_zone_plate/depends_on-field`
- `/NXfresnel_zone_plate/fabrication-field`
- `/NXfresnel_zone_plate/focus_parameters-field`
- `/NXfresnel_zone_plate/mask_material-field`
- `/NXfresnel_zone_plate/mask_thickness-field`
- `/NXfresnel_zone_plate/outer_diameter-field`
- `/NXfresnel_zone_plate/outermost_zone_width-field`
- `/NXfresnel_zone_plate/support_membrane_material-field`
- `/NXfresnel_zone_plate/support_membrane_thickness-field`
- `/NXfresnel_zone_plate/TRANSFORMATIONS-group`
- `/NXfresnel_zone_plate/zone_height-field`
- `/NXfresnel_zone_plate/zone_material-field`
- `/NXfresnel_zone_plate/zone_support_material-field`
- `/NXfresnel_zone_plate@default-attribute`

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/base_classes/NXfresnel_zone_plate.nxdl.xml
NXgeometry

Status:

base class, extends NXobject

DEPRECATED: as decided at 2014 NIAC meeting, convert to use NXtransformations

Description:

legacy class - recommend to use NXtransformations now

It is recommended that instances of NXgeometry be converted to use NXtransformations.

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.

Symbols:

No symbol table

Groups cited:

NXorientation, NXshape, NXtranslation

Structure:

@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

description: (optional) 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: (optional) 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.

SHAPE: (optional) NXshape

shape/size information of component

TRANSLATION: (optional) NXtranslation

translation of component

ORIENTATION: (optional) NXorientation

orientation of component
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXgeometry/component_index-field
- /NXgeometry/description-field
- /NXgeometry/ORIENTATION-group
- /NXgeometry/SHAPE-group
- /NXgeometry/TRANSLATION-group
- /NXgeometry@default-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXgeometry.nxdl.xml

NXgrating

Status:
base class, extends NXobject

Description:
A diffraction grating, as could be used in a soft X-ray monochromator

Symbols:
No symbol table

Groups cited:
NXdata, NXoff_geometry, NXshape, NXtransformations

Structure:

@default: (optional) NX_CHAR
Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default
dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html
for a summary of the discussion.

angles: (optional) NX_FLOAT (Rank: 1, Dimensions: [2]) {units=NX_ANGLE}
Blaze or trapezoidal angles, with the angle of the upstream facing edge listed first. Blazed
gratings can be identified by the low value of the first-listed angle.

period: (optional) NX_FLOAT (Rank: 1) {units=NX_LENGTH}
List of polynomial coefficients describing the spatial separation of lines/grooves as a function
of position along the grating, in increasing powers of position. Gratings which do not have
variable line spacing will only have a single coefficient (constant).

duty_cycle: (optional) NX_FLOAT {units=NX_UNITLESS}
depth: (optional) NX_FLOAT {units=NX_LENGTH}
diffraction_order: (optional) NX_INT {units=NX_UNITLESS}
deflection_angle: (optional) NX_FLOAT {units=NX_ANGLE}
Angle between the incident beam and the utilised outgoing beam.

3.3. NeXus Class Definitions 241
interior_atmosphere: (optional) **NX_CHAR**

Any of these values: vacuum | helium | argon

substrate_material: (optional) **NX_CHAR**

substrate_density: (optional) **NX_FLOAT** {units=NX_MASS_DENSITY}

substrate_thickness: (optional) **NX_FLOAT** {units=NX_LENGTH}

coating_material: (optional) **NX_CHAR**

substrate_roughness: (optional) **NX_FLOAT** {units=NX_LENGTH}

coating_roughness: (optional) **NX_FLOAT** {units=NX_LENGTH}

layer_thickness: (optional) **NX_FLOAT** {units=NX_LENGTH}

An array describing the thickness of each layer

depends_on: (optional) **NX_CHAR**

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

shape: (optional) **NXshape**

DEPRECATED: Use NXoff_geometry to describe the shape of grating

A NXshape group describing the shape of the mirror

figure_data: (optional) **NXdata**

Numerical description of the surface figure of the mirror.

OFF_GEOMETRY: (optional) **NXoff_geometry**

This group describes the shape of the beam line component

TRANSFORMATIONS: (optional) **NXtransformations**

“Engineering” position of the grating Transformations used by this component to define its position and orientation.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXgrating/angles-field
- /NXgrating/coating_material-field
- /NXgrating/coating_roughness-field
- /NXgrating/deflection_angle-field
- /NXgrating/depends_on-field
- /NXgrating/depth-field
- /NXgrating/diffraction_order-field
- /NXgrating/duty_cycle-field
NXguide

Status:

base class, extends NXobject

Description:

A neutron optical element to direct the path of the beam.

NXguide is used by neutron instruments to describe a guide consists of several mirrors building a shape through which neutrons can be guided or directed. The simplest such form is box shaped although elliptical guides are gaining in popularity. The individual parts of a guide usually have common characteristics but there are cases where they are different. For example, a neutron guide might consist of 2 or 4 coated walls or a supermirror bender with multiple, coated vanes.

To describe polarizing supermirrors such as used in neutron reflection, it may be necessary to revise this definition of NXguide to include NXpolarizer and/or NXmirror.

When even greater complexity exists in the definition of what constitutes a guide, it has been suggested that NXguide be redefined as a NXcollection of NXmirror each having their own NXgeometry describing their location(s).

For the more general case when describing mirrors, consider using NXmirror.

NOTE: The NeXus International Advisory Committee welcomes comments for revision and improvement of this definition of NXguide.

Symbols:

nsurf: number of reflecting surfaces

nwl: number of wavelengths

Groups cited:

NXdata, NXgeometry, NXoff_geometry, NXtransformations

Structure:
@default: (optional) \textit{NX_CHAR}

Declares which child group contains a path leading to a \textit{NXdata} group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

description: (optional) \textit{NX_CHAR}

A description of this particular instance of \texttt{NXguide}.

\textbf{incident\_angle:} (optional) \textit{NX_FLOAT} \{units=\textit{NX\_ANGLE}\}

TODO: documentation needed

\textbf{bend\_angle\_x:} (optional) \textit{NX_FLOAT} \{units=\textit{NX\_ANGLE}\}

TODO: documentation needed

\textbf{bend\_angle\_y:} (optional) \textit{NX_FLOAT} \{units=\textit{NX\_ANGLE}\}

TODO: documentation needed

\textbf{interior\_atmosphere:} (optional) \textit{NX_CHAR}

Any of these values: \texttt{vacuum} | \texttt{helium} | \texttt{argon}

\textbf{external\_material:} (optional) \textit{NX_CHAR}

e external material outside substrate

\textbf{m\_value:} (optional) \textit{NX_FLOAT} (Rank: 1, Dimensions: \{nsurf\})

The \texttt{m} value for a supermirror, which defines the supermirror regime in multiples of the critical angle of Nickel.

\textbf{substrate\_material:} (optional) \textit{NX_FLOAT} (Rank: 1, Dimensions: \{nsurf\})

TODO: documentation needed

\textbf{substrate\_thickness:} (optional) \textit{NX_FLOAT} (Rank: 1, Dimensions: \{nsurf\}) \{units=\textit{NX\_LENGTH}\}

TODO: documentation needed

\textbf{coating\_material:} (optional) \textit{NX_FLOAT} (Rank: 1, Dimensions: \{nsurf\})

TODO: documentation needed

\textbf{substrate\_roughness:} (optional) \textit{NX_FLOAT} (Rank: 1, Dimensions: \{nsurf\}) \{units=\textit{NX\_LENGTH}\}

TODO: documentation needed

\textbf{coating\_roughness:} (optional) \textit{NX_FLOAT} (Rank: 1, Dimensions: \{nsurf\}) \{units=\textit{NX\_LENGTH}\}

TODO: documentation needed

\textbf{number\_sections:} (optional) \textit{NX_INT} \{units=\texttt{NX\_UNITLESS}\}

number of substrate sections (also called \texttt{nsurf} as an index in the \texttt{NXguide} specification)

\textbf{depends\_on:} (optional) \textit{NX_CHAR}

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The \texttt{depends\_on} field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.
The entry opening of the guide lies on the reference plane. The center of the opening on that plane is the reference point on the x and y axis. The reference plane is orthogonal to the z axis and is the reference point along the z axis. Given no bend in the guide, it is parallel with z axis and extends in the positive direction of the z axis.

GEOMETRY: (optional) \textit{NXgeometry}

\textbf{DEPRECATED}: Use the field \textit{depends_on} and \textit{NXtransformations} to position the guid and \textit{NXoff\_geometry} to describe its shape instead

\textbf{TODO}: Explain what this NXgeometry group means. What is intended here?

\textbf{reflectivity}: (optional) \textit{NXdata}

Reflectivity as function of reflecting surface and wavelength

\textbf{@signal}: (optional) \textit{NX\_CHAR}

Obligatory value: data

\textbf{@axes}: (optional) \textit{NX\_CHAR}

Obligatory value: surface wavelength

\textbf{@surface\_indices}: (optional) \textit{NX\_CHAR}

Obligatory value: 0

\textbf{@wavelength\_indices}: (optional) \textit{NX\_CHAR}

Obligatory value: 1

\textbf{data}: (optional) \textit{NX\_NUMBER} (Rank: 2, Dimensions: [nsurf, nwl])

reflectivity of each surface as a function of wavelength

\textbf{surface}: (optional) \textit{NX\_NUMBER} (Rank: 1, Dimensions: [nsurf]) \{units=\textit{NX\_ANY}\}

List of surfaces. Probably best to use index numbers but the specification is very loose.

\textbf{wavelength}: (optional) \textit{NX\_NUMBER} (Rank: 1, Dimensions: [nwl]) \{units=\textit{NX\_WAVELENGTH}\}

wavelengths at which reflectivity was measured

\textbf{OFF\_GEOMETRY}: (optional) \textit{NXoff\_geometry}

This group describes the shape of the beam line component

\textbf{TRANSFORMATIONS}: (optional) \textit{NXtransformations}

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXguide/bend_angle_x-field
- /NXguide/bend_angle_y-field
- /NXguide/coating_material-field
- /NXguide/coating_roughness-field
- /NXguide/depends_on-field
- /NXguide/description-field
- /NXguide/external_material-field
- /NXguide/GEOMETRY-group
- /NXguide/incident_angle-field
- /NXguide/interior_atmosphere-field
- /NXguide/m_value-field
- /NXguide/number_sections-field
- /NXguide/OFF_GEOMETRY-group
- /NXguide/reflectivity-group
- /NXguide/reflectivity/data-field
- /NXguide/reflectivity/surface-field
- /NXguide/reflectivity/wavelength-field
- /NXguide/reflectivity@axes-attribute
- /NXguide/reflectivity@signal-attribute
- /NXguide/reflectivity@surface_indices-attribute
- /NXguide/reflectivity@wavelength_indices-attribute
- /NXguide/substrate_material-field
- /NXguide/substrate_roughness-field
- /NXguide/substrate_thickness-field
- /NXguide/TRANSFORMATIONS-group
- /NXguide@default-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXguide.nxdl.xml
**NXinsertion_device**

**Status:**
base class, extends `NXobject`

**Description:**
An insertion device, as used in a synchrotron light source.

**Symbols:**
No symbol table

**Groups cited:**
`NXdata`, `NXgeometry`, `NXoff_geometry`, `NXtransformations`

**Structure:**

@default: (optional) `NX_CHAR`
Declares which child group contains a path leading to a `NXdata` group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

type: (optional) `NX_CHAR`
Any of these values: undulator | wiggler

gap: (optional) `NX_FLOAT` {units=`NX_LENGTH`}
separation between opposing pairs of magnetic poles

taper: (optional) `NX_FLOAT` {units=`NX_ANGLE`}
angular of gap difference between upstream and downstream ends of the insertion device

phase: (optional) `NX_FLOAT` {units=`NX_ANGLE`}

poles: (optional) `NX_INT` {units=`NX_UNITLESS`}
number of poles

magnetic_wavelength: (optional) `NX_FLOAT` {units=`NX_WAVELENGTH`}

k: (optional) `NX_FLOAT` {units=`NX_DIMENSIONLESS`}
beam displacement parameter

length: (optional) `NX_FLOAT` {units=`NX_LENGTH`}
length of insertion device

power: (optional) `NX_FLOAT` {units=`NX_POWER`}
total power delivered by insertion device

energy: (optional) `NX_FLOAT` {units=`NX_ENERGY`}
energy of peak intensity in output spectrum

bandwidth: (optional) `NX_FLOAT` {units=`NX_ENERGY`}
bandwidth of peak energy

harmonic: (optional) `NX_INT` {units=`NX_UNITLESS`}
harmonic number of peak
depends_on: (optional) NX_CHAR

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

spectrum: (optional) NXdata

spectrum of insertion device

GEOMETRY: (optional) NXgeometry

DEPRECATED: Use the field depends_on and NXtransformations to position the device and NXoff_geometry to describe its shape instead

“Engineering” position of insertion device

OFF_GEOMETRY: (optional) NXoff_geometry

This group describes the shape of the beam line component

TRANSFORMATIONS: (optional) NXtransformations

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

• /NXinsertion_device/bandwidth-field
• /NXinsertion_device/depends_on-field
• /NXinsertion_device/energy-field
• /NXinsertion_device/gap-field
• /NXinsertion_device/GEOMETRY-group
• /NXinsertion_device/harmonic-field
• /NXinsertion_device/k-field
• /NXinsertion_device/length-field
• /NXinsertion_device/magnetic_wavelength-field
• /NXinsertion_device/OFF_GEOMETRY-group
• /NXinsertion_device/phase-field
• /NXinsertion_device/poles-field
• /NXinsertion_device/power-field
• /NXinsertion_device/spectrum-group
• /NXinsertion_device/taper-field
• /NXinsertion_device/TRANSFORMATIONS-group
• /NXinsertion_device/type-field
NXinstrument

Status:

base class, extends NXobject

Description:

Collection of the components of the instrument or beamline.

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.

Symbols:

No symbol table

Groups cited:

NXaperture, NXattenuator, NXbeam_stop, NXbeam, NXbending_magnet, NXcapillary, NXcollection, NXcollimator, NXcrystal, NXdetector_group, NXdetector, NXdisk_chopper, NXevent_data, NXfermi_chopper, NXfilter, NXflipper, NXguide, NXinsertion_device, NXmirror, NXmoderator, NXmonochromator, NXpolarizer, NXPoisitioner, NXsource, NXtransformations, NXvelocity_selector, NXxraylens

Structure:

@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

name: (optional) NX_CHAR

Name of instrument

@short_name: (optional) NX_CHAR

short name for instrument, perhaps the acronym

APERTURE: (optional) NXaperture

ATTENUATOR: (optional) NXattenuator

BEAM: (optional) NXbeam

BEAM_STOP: (optional) NXbeam_stop

BENDING_MAGNET: (optional) NXbending_magnet

COLLIMATOR: (optional) NXcollimator

COLLECTION: (optional) NXcollection

CAPILLARY: (optional) NXcapillary

CRYSTAL: (optional) NXcrystal
DETECTOR: (optional) NXdetector
DETECTOR_GROUP: (optional) NXdetector_group
DISK_CHOPPER: (optional) NXdisk_chopper
EVENT_DATA: (optional) NXevent_data
FERMI_CHOPPER: (optional) NXfermi_chopper
FILTER: (optional) NXfilter
FLIPPER: (optional) NXflipper
GUIDE: (optional) NXguide
INSERTION_DEVICE: (optional) NXinsertion_device
MIRROR: (optional) NXmirror
MODERATOR: (optional) NXmoderator
MONOCHROMATOR: (optional) NXmonochromator
POLARIZER: (optional) NXpolarizer
POSITIONER: (optional) NXpositioner
SOURCE: (optional) NXsource
DIFFRACTOMETER: (optional) NXtransformations
VELOCITY_SELECTOR: (optional) NXvelocity_selector
XRAYLENS: (optional) NXxraylens

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXinstrument/APERTURE-group
- /NXinstrument/ATTENUATOR-group
- /NXinstrument/BEAM-group
- /NXinstrument/BEAM_STOP-group
- /NXinstrument/BENDING_MAGNET-group
- /NXinstrument/CAPILLARY-group
- /NXinstrument/COLLECTION-group
- /NXinstrument/COLLIMATOR-group
- /NXinstrument/CRYSTAL-group
- /NXinstrument/DETECTOR-group
- /NXinstrument/DETECTOR_GROUP-group
- /NXinstrument/DIFFRACTOMETER-group
- /NXinstrument/DISK_CHOPPER-group
- /NXinstrument/EVENT_DATA-group
- /NXinstrument/FERMI_CHOPPER-group
**/NXinstrument/FILTER-group**

**/NXinstrument/FLIPPER-group**

**/NXinstrument/GUIDE-group**

**/NXinstrument/INSERTION_DEVICE-group**

**/NXinstrument/MIRROR-group**

**/NXinstrument/MODERATOR-group**

**/NXinstrument/MONOCHROMATOR-group**

**/NXinstrument/name-field**

**/NXinstrument/name@short_name-attribute**

**/NXinstrument/POLARIZER-group**

**/NXinstrument/POSITIONER-group**

**/NXinstrument/SOURCE-group**

**/NXinstrument/VELOCITY_SELECTOR-group**

**/NXinstrument/XRAYLENS-group**

**/NXinstrument@default-attribute**

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/base_classes/NXinstrument.nxdl.xml

### NXlog

**Status:**

base class, extends **NXobject**

**Description:**

Information recorded as a function of time.

Description of information that is recorded against time. There are two common use cases for this:

- When logging data such as temperature during a run
- When data is taken in streaming mode data acquisition, i.e. just timestamp, value pairs are stored and correlated later in data reduction with other data,

In both cases, NXlog contains the logged or streamed values and the times at which they were measured as elapsed time since a starting time recorded in ISO8601 format. The time units are specified in the units attribute. An optional scaling attribute can be used to accommodate non standard clocks.

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.

In order to make random access to timestamped data faster there is an optional array pair of cue_timestamp_zero and cue_index. The cue_timestamp_zero will contain coarser timestamps than in the time array, say every five minutes. The cue_index will then contain the index into the time,value pair of arrays for that coarser cue_timestamp_zero.

**Symbols:**

No symbol table
Groups cited:
none

Structure:

@default: (optional) **NX_CHAR**

Declares which child group contains a path leading to a *NXdata* group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

time: (optional) **NX_NUMBER** \{units=*NX_TIME*\}

Time of logged entry. The times are relative to the “start” attribute and in the units specified in the “units” attribute. Please note that absolute timestamps under unix are relative to 1970-01-01T00:00:00.0Z.

The scaling_factor, when present, has to be applied to the time values in order to arrive at the units specified in the units attribute. The scaling_factor allows for arbitrary time units such as ticks of some hardware clock.

@start: (optional) **NX_DATE_TIME**

@scaling_factor: (optional) **NX_NUMBER**

value: (optional) **NX_NUMBER** \{units=*NX_ANY*\}

Array of logged value, such as temperature. If this is a single value the dimensionality is nEntries. However, NXlog can also be used to store multi dimensional time stamped data such as images. In this example the dimensionality of values would be value[nEntries,xdim,ydim].

raw_value: (optional) **NX_NUMBER** \{units=*NX_ANY*\}

Array of raw information, such as thermocouple voltage

description: (optional) **NX_CHAR**

Description of logged value

average_value: (optional) **NX_FLOAT** \{units=*NX_ANY*\}

average_value_error: (optional) **NX_FLOAT** \{units=*NX_ANY*\}

DEPRECATED: see: https://github.com/nexusformat/definitions/issues/639

estimated uncertainty (often used: standard deviation) of average_value

average_value_errors: (optional) **NX_FLOAT** \{units=*NX_ANY*\}

estimated uncertainty (often used: standard deviation) of average_value

minimum_value: (optional) **NX_FLOAT** \{units=*NX_ANY*\}

maximum_value: (optional) **NX_FLOAT** \{units=*NX_ANY*\}

duration: (optional) **NX_FLOAT** \{units=*NX_ANY*\}

Total time log was taken

cue_timestamp_zero: (optional) **NX_NUMBER** \{units=*NX_TIME*\}

Timestamps matching the corresponding cue_index into the time, value pair.

@start: (optional) **NX_DATE_TIME**

If missing start is assumed to be the same as for “time”.

---

252 Chapter 3. NeXus: Reference Documentation
@scaling_factor: (optional) $NX\_NUMBER$

If missing start is assumed to be the same as for “time”.

cue_index: (optional) $NX\_INT$

Index into the time, value pair matching the corresponding cue_timestamp_zero.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXlog/average_value-field
- /NXlog/average_value_error-field
- /NXlog/average_value_errors-field
- /NXlog/cue_index-field
- /NXlog/cue_timestamp_zero-field
- /NXlog/cue_timestamp_zero@scaling_factor-attribute
- /NXlog/cue_timestamp_zero@start-attribute
- /NXlog/description-field
- /NXlog/duration-field
- /NXlog/maximum_value-field
- /NXlog/minimum_value-field
- /NXlog/raw_value-field
- /NXlog/time-field
- /NXlog/time@scaling_factor-attribute
- /NXlog/time@start-attribute
- /NXlog/value-field
- /NXlog@default-attribute

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/base_classes/NXlog.nxdl.xml

**NXmirror**

**Status:**

base class, extends $NXobject$

**Description:**

A beamline mirror or supermirror.

**Symbols:**

No symbol table

**Groups cited:**

$NXdata$, $NXgeometry$, $NXoff\_geometry$, $NXshape$, $NXtransformations$
Structure:

@default: (optional) \textit{NX_CHAR}

Declares which child group contains a path leading to a \textit{NXdata} group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

type: (optional) \textit{NX_CHAR}

Any of these values:

- single: mirror with a single material as a reflecting surface
- multi: mirror with stacked, multiple layers as a reflecting surface

description: (optional) \textit{NX_CHAR}

description of this mirror

incident_angle: (optional) \textit{NX_FLOAT} \{units=\textit{NX_ANGLE}\}

bend_angle_x: (optional) \textit{NX_FLOAT} \{units=\textit{NX_ANGLE}\}

bend_angle_y: (optional) \textit{NX_FLOAT} \{units=\textit{NX_ANGLE}\}

interior_atmosphere: (optional) \textit{NX_CHAR}

Any of these values: vacuum | helium | argon

external_material: (optional) \textit{NX_CHAR}

external material outside substrate

m_value: (optional) \textit{NX_FLOAT} \{units=\textit{NX_UNITLESS}\}

The m value for a supermirror, which defines the supermirror regime in multiples of the critical angle of Nickel.

substrate_material: (optional) \textit{NX_CHAR}

substrate_density: (optional) \textit{NX_FLOAT} \{units=\textit{NX_MASS_DENSITY}\}

substrate_thickness: (optional) \textit{NX_FLOAT} \{units=\textit{NX_LENGTH}\}

coating_material: (optional) \textit{NX_CHAR}

substrate_roughness: (optional) \textit{NX_FLOAT} \{units=\textit{NX_LENGTH}\}

coating_roughness: (optional) \textit{NX_FLOAT} \{units=\textit{NX_LENGTH}\}

even_layer_material: (optional) \textit{NX_CHAR}

even_layer_density: (optional) \textit{NX_FLOAT} \{units=\textit{NX_MASS_DENSITY}\}

odd_layer_material: (optional) \textit{NX_CHAR}

odd_layer_density: (optional) \textit{NX_FLOAT} \{units=\textit{NX_MASS_DENSITY}\}

layer_thickness: (optional) \textit{NX_FLOAT} \{units=\textit{NX_LENGTH}\}

An array describing the thickness of each layer

depends_on: (optional) \textit{NX_CHAR}
NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “." if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

Given a flat mirror, the reference plane is the plane which contains the “entry” surface of the mirror. The reference point of the mirror in the x and y axis is the centre of the mirror on that plane. The reference plane is orthogonal to the z axis and the location of this plane is the reference point on the z axis. The mirror faces negative z values.

![Diagram of a mirror with reference plane and axes](image)

**GEOMETRY:** (optional) *NXgeometry*

**DEPRECATED:** Use the field *depends_on* and *NXtransformations* to position the mirror and *NXoff_geometry* to describe its shape instead.

**reflectivity:** (optional) *NXdata*

Reflectivity as function of wavelength

**shape:** (optional) *NXshape*

**DEPRECATED:** Use *NXoff_geometry* instead.

A NXshape group describing the shape of the mirror

**figure_data:** (optional) *NXdata*

Numerical description of the surface figure of the mirror.

**OFF_GEOMETRY:** (optional) *NXoff_geometry*

This group describes the shape of the beam line component

**TRANSFORMATIONS:** (optional) *NXtransformations*

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

### Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXmirror/bend_angle_x-field
- /NXmirror/bend_angle_y-field
- /NXmirror/coating_material-field
- /NXmirror/coating_roughness-field
- /NXmirror/depends_on-field
- /NXmirror/description-field
- /NXmirror/even_layer_density-field
NXmirror

- /NXmirror/even_layer_material-field
- /NXmirror/external_material-field
- /NXmirror/figure_data-group
- /NXmirror/GEOMETRY-group
- /NXmirror/incident_angle-field
- /NXmirror/interior_atmosphere-field
- /NXmirror/layer_thickness-field
- /NXmirror/m_value-field
- /NXmirror/odd_layer_density-field
- /NXmirror/odd_layer_material-field
- /NXmirror/OFF_GEOMETRY-group
- /NXmirror/reflectivity-group
- /NXmirror/shape-group
- /NXmirror/substrate_density-field
- /NXmirror/substrate_material-field
- /NXmirror/substrate_roughness-field
- /NXmirror/substrate_thickness-field
- /NXmirror/TRANSFORMATIONS-group
- /NXmirror/type-field
- /NXmirror@default-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXmirror.nxdl.xml

NXmoderator

Status:
- base class, extends NXobject

Description:
- A neutron moderator

Symbols:
- No symbol table

Groups cited:
- NXdata, NXgeometry, NXlog, NXoff_geometry, NXtransformations

Structure:
- @default: (optional) NX_CHAR
Declares which child group contains a path leading to a \textit{NXdata} group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

\textbf{distance}: (optional) \texttt{NX_FLOAT \{units=NX\_LENGTH\}}

Effective distance as seen by measuring radiation. Note, it is recommended to use NXtransformations instead.

\textbf{type}: (optional) \texttt{NX\_CHAR}

Any of these values:

- H2O
- D2O
- Liquid H2
- Liquid CH4
- Liquid D2
- Solid D2
- C
- Solid CH4
- Solid H2

\textbf{poison_depth}: (optional) \texttt{NX\_FLOAT \{units=NX\_LENGTH\}}

\textbf{coupled}: (optional) \texttt{NX\_BOOLEAN}

whether the moderator is coupled

\textbf{coupling_material}: (optional) \texttt{NX\_CHAR}

The material used for coupling. Usually Cd.

\textbf{poison_material}: (optional) \texttt{NX\_CHAR}

Any of these values: Gd | Cd

\textbf{temperature}: (optional) \texttt{NX\_FLOAT \{units=NX\_TEMPERATURE\}}

average/nominal moderator temperature

\textbf{depends_on}: (optional) \texttt{NX\_CHAR}

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The \texttt{depends\_on} field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

The reference point of the moderator is its center in the x and y axis. The reference point on the z axis is the surface of the moderator pointing towards the source (the negative part of the z axis).
GEOMETRY: (optional) **NXgeometry**

**DEPRECATED**: Use the field `depends_on` and `NXtransformations` to position the moderator and `NXoff_geometry` to describe its shape instead

“Engineering” position of moderator

**temperature_log**: (optional) **NXlog**

log file of moderator temperature

**pulse_shape**: (optional) **NXdata**

moderator pulse shape

**OFF_GEOMETRY**: (optional) **NXoff_geometry**

This group describes the shape of the moderator

**TRANSFORMATIONS**: (optional) **NXtransformations**

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXmoderator/coupled-field
- /NXmoderator/coupling_material-field
- /NXmoderator/depends_on-field
- /NXmoderator/distance-field
- /NXmoderator/GEOMETRY-group
- /NXmoderator/OFF_GEOMETRY-group
- /NXmoderator/poison_depth-field
- /NXmoderator/poison_material-field
- /NXmoderator/pulse_shape-group
- /NXmoderator/temperature-field
- /NXmoderator/temperature_log-group
- /NXmoderator/TRANSFORMATIONS-group
- /NXmoderator/type-field
- /NXmoderator@default-attribute

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/base_classes/NXmoderator.nxdl.xml
**NXmonitor**

**Status:**

base class, extends *NXobject*

**Description:**

A monitor of incident beam 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.

**Symbols:**

No symbol table

**Groups cited:**

*NXgeometry, NXlog, NXoff_geometry, NXtransformations*

**Structure:**

@default: (optional) *NX_CHAR*

Declares which child group contains a path leading to a *NXdata* group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See [https://www.nexusformat.org/2014_How_to_find_default_data.html](https://www.nexusformat.org/2014_How_to_find_default_data.html) for a summary of the discussion.

mode: (optional) *NX_CHAR*

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: monitor | timer

start_time: (optional) *NX_DATE_TIME*

Starting time of measurement

der_time: (optional) *NX_DATE_TIME*

Ending time of measurement

preset: (optional) *NX_NUMBER {units=NX_ANY}*

preset value for time or monitor

distance: (optional) *NX_FLOAT {units=NX_LENGTH}*

**DEPRECATED:** Use transformations/distance instead

Distance of monitor from sample

range: (optional) *NX_FLOAT {Rank: 1, Dimensions: [2]} {units=NX_ANY}*

Range (X-axis, Time-of-flight, etc.) over which the integral was calculated

nominal: (optional) *NX_NUMBER {units=NX_ANY}*

Nominal reading to be used for normalisation purposes.

integral: (optional) *NX_NUMBER {units=NX_ANY}*

Total integral monitor counts

type: (optional) *NX_CHAR*
Any of these values: Fission Chamber | Scintillator

time_of_flight: (optional) NX_FLOAT (Rank: same as field efficiency, Dimensions: same as field efficiency) {units=NX_TIME_OF_FLIGHT}

Time-of-flight

efficiency: (optional) NX_NUMBER (Rank: same as field i, Dimensions: same as field i) {units=NX_DIMENSIONLESS}

Monitor efficiency

data: (optional) NX_NUMBER (Rank: dataRank) {units=NX_ANY}

Monitor data

sampled_fraction: (optional) NX_FLOAT {units=NX_DIMENSIONLESS}

Proportion of incident beam sampled by the monitor (0<x<1)

count_time: (optional) NX_FLOAT {units=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

depends_on: (optional) NX_CHAR

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

The reference plane of the monitor contains the surface of the detector that faces the source and is the entry point of the beam. The reference point of the monitor in the x and y axis is its centre on this surface. The reference plane is orthogonal to the the z axis and the reference point on this z axis is where they intersect.

integral_log: (optional) NXlog

Time variation of monitor counts

GEOMETRY: (optional) NXgeometry

DEPRECATED: Use the field depends_on and NXtransformations to position the monitor and NXoff_geometry to describe its shape instead

Geometry of the monitor

OFF_GEOMETRY: (optional) NXoff_geometry

This group describes the shape of the beam line component

TRANSFORMATIONS: (optional) NXtransformations

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXmonitor/count_time-field
- /NXmonitor/data-field
- /NXmonitor/depends_on-field
- /NXmonitor/distance-field
- /NXmonitor/efficiency-field
- /NXmonitor/end_time-field
- /NXmonitor/GEOMETRY-group
- /NXmonitor/integral-field
- /NXmonitor/integral_log-group
- /NXmonitor/mode-field
- /NXmonitor/nominal-field
- /NXmonitor/OFF_GEOMETRY-group
- /NXmonitor/preset-field
- /NXmonitor/range-field
- /NXmonitor/sampled_fraction-field
- /NXmonitor/start_time-field
- /NXmonitor/time_of_flight-field
- /NXmonitor/TRANSFORMATIONS-group
- /NXmonitor/type-field
- /NXmonitor@default-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXmonitor.nxdl.xml

NXmonochromator

Status:
base class, extends NXobject

Description:
A wavelength defining device.
This is a base class for everything which selects a wavelength or energy, be it a monochromator crystal, a velocity selector, an undulator or whatever.
The expected units are:
- wavelength: angstrom
- energy: eV

Symbols:
No symbol table

Groups cited:
NXcrystal, NXdata, NXgeometry, NXgrating, NXoff_geometry, NXtransformations, NXvelocity_selector

Structure:

@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

wavelength: (optional) NX_FLOAT {units=NX_WAVELENGTH}

wavelength selected

wavelength_error: (optional) NX_FLOAT {units=NX_WAVELENGTH}

DEPRECATED: see https://github.com/nexusformat/definitions/issues/820

wavelength standard deviation

wavelength_errors: (optional) NX_FLOAT {units=NX_WAVELENGTH}

wavelength standard deviation

energy: (optional) NX_FLOAT {units=NX_ENERGY}

energy selected

energy_error: (optional) NX_FLOAT {units=NX_ENERGY}

DEPRECATED: see https://github.com/nexusformat/definitions/issues/820

energy standard deviation

energy_errors: (optional) NX_FLOAT {units=NX_ENERGY}

energy standard deviation

depends_on: (optional) NX_CHAR

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

distribution: (optional) NXdata

group: (optional) NXgeometry

DEPRECATED: Use the field depends_on and NXtransformations to position the monochromator and NXoff_geometry to describe its shape instead

OFF_GEOMETRY: (optional) NXoff_geometry

This group describes the shape of the beam line component

CRYSTAL: (optional) NXcrystal

Use as many crystals as necessary to describe

VELOCITY_SELECTOR: (optional) NXvelocity_selector

GRATING: (optional) NXgrating
For diffraction grating based monochromators

**TRANSFORMATIONS**: (optional) *NXtransformations*

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

### Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXmonochromator/CRYSTAL-group`
- `/NXmonochromator/depends_on-field`
- `/NXmonochromator/distribution-group`
- `/NXmonochromator/energy-field`
- `/NXmonochromator/energy_error-field`
- `/NXmonochromator/energy_errors-field`
- `/NXmonochromator/geometry-group`
- `/NXmonochromator/GRATING-group`
- `/NXmonochromator/OFF_GEOMETRY-group`
- `/NXmonochromator/TRANSFORMATIONS-group`
- `/NXmonochromator/VELOCITY_SELECTOR-group`
- `/NXmonochromator/wavelength-field`
- `/NXmonochromator/wavelength_error-field`
- `/NXmonochromator/wavelength_errors-field`
- `/NXmonochromator@default-attribute`

**NXDL Source:**

**NXnote**

**Status:**
base class, extends *NXobject*

**Description:**
Any additional freeform information not covered by the other base classes.

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

**Symbols:**
No symbol table

**Groups cited:**
none
Structure:

@default: (optional) NX_CHAR
Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

author: (optional) NX_CHAR
Author or creator of note

date: (optional) NX_DATE_TIME
Date note created/added

type: (optional) NX_CHAR
Mime content type of note data field e.g. image/jpeg, text/plain, text/html

file_name: (optional) NX_CHAR
Name of original file name if note was read from an external source

description: (optional) NX_CHAR
Title of an image or other details of the note

sequence_index: (optional) NX_POSINT
Sequence index of note, for placing a sequence of multiple NXnote groups in an order. Starts with 1.

data: (optional) NX_BINARY
Binary note data - if text, line terminator is [CR][LF].

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

• /NXnote/author-field
• /NXnote/data-field
• /NXnote/date-field
• /NXnote>Description-field
• /NXnote/file_name-field
• /NXnote/sequence_index-field
• /NXnote/type-field
• /NXnote@default-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXnote.nxdl.xml
**NXobject**

**Status:**
- base class, extends none

**Description:**
- This is the base object of NeXus

**Symbols:**
- No symbol table

**Groups cited:**
- none

**Structure:**

**NXDL Source:**
- https://github.com/nexusformat/definitions/blob/main/base_classes/NXobject.nxdl.xml

**NXoff_geometry**

**Status:**
- base class, extends *NXobject*

**Description:**
- Geometry (shape) description. The format closely matches the Object File Format (OFF) which can be output by most CAD software. It can be used to describe the shape of any beamline component, including detectors. In the case of detectors it can be used to define the shape of a single pixel, or, if the pixel shapes are non-uniform, to describe the shape of the whole detector.

**Symbols:**
- These symbols will be used below.
  - i: number of vertices in the shape
  - k: number of faces in the shape
  - l: number faces which are detecting surfaces or form the boundary of detecting volumes

**Groups cited:**
- none

**Structure:**

- @default: (optional) *NX_CHAR*
  
  Declares which child group contains a path leading to a *NXdata* group.

  It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

- vertices: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [i, 3]) \{units=NX_LENGTH\}
  
  List of x,y,z coordinates for vertices. The origin of the coordinates is the position of the parent component, for example the NXdetector which the geometry describes. If the shape describes a single pixel for a detector with uniform pixel shape then the origin is the position of each pixel as described by the x/y/z_pixel_offset datasets in NXdetector.
**winding_order**: (optional) *NX_INT* (Rank: 1, Dimensions: [j])

List of indices of vertices in the vertices dataset to form each face, right-hand rule for face normal.

**faces**: (optional) *NX_INT* (Rank: 1, Dimensions: [k])

The start index in winding_order for each face.

**detector_faces**: (optional) *NX_INT* (Rank: 2, Dimensions: [l, 2])

List of pairs of index in the “faces” dataset and detector id. Face IDs in the first column, and corresponding detector IDs in the second column. This dataset should only be used only if the NXoff_geometry group is describing a detector. Note, the face indices must be in ascending order but need not be consecutive as not every face in faces need be a detecting surface or boundary of detecting volume. Can use multiple entries with the same detector id to define detector volumes.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXoff_geometry/detector_faces-field
- /NXoff_geometry/faces-field
- /NXoff_geometry/vertices-field
- /NXoff_geometry/winding_order-field
- /NXoff_geometry@default-attribute

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/base_classes/NXoff_geometry.nxdl.xml

**NXorientation**

**Status:**
base class, extends *NXObject*

**Description:**
legacy class - recommend to use *NXtransformations* now
Description for a general orientation of a component - used by *NXgeometry*

**Symbols:**
No symbol table

**Groups cited:**
*NXgeometry*

**Structure:**

- [@default]: (optional) *NX_CHAR*

Declares which child group contains a path leading to a *NXdata* group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.
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 \([x' \cdot x, x' \cdot y, x' \cdot z, y' \cdot x, y' \cdot y, y' \cdot z]\) 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.

The pair of groups NXtranslation and NXorientation together describe the position of a component.

**GEOMETRY**: (optional) **NXgeometry**

Link to another object if we are using relative positioning, else absent

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXorientation/GEOmetry-group
- /NXorientation/value-field
- /NXorientation@default-attribute

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/base_classes/NXorientation.nxdl.xml

**NXparameters**

**Status:**

Base class, extends **NXobject**

**Description:**

Container for parameters, usually used in processing or analysis.

**Symbols:**

No symbol table

**Groups cited:**

none

**Structure:**

@**default**: (optional) **NX_CHAR**

Declares which child group contains a path leading to a **NXdata** group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

**term**: (optional) **NX_CHAR**

A parameter (also known as a term) that is used in or results from processing.

@**units**: (optional) **NX_CHAR**
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXparameters/term-field`
- `/NXparameters/term@units-attribute`
- `/NXparameters@default-attribute`

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXparameters.nxdl.xml

NXpdb

Status:
base class, extends NXobject

Description:
A NeXus transliteration of a PDB file, to be validated only as a PDB rather than in NeXus. Use NXpdb to incorporate the information in an arbitrary PDB into a NeXus file.

The main suggestion is to use this as a container class for a PDB entry to describe a sample in NXsample, but it may be more appropriate to place this higher in the hierarchy, say in NXentry.

The structure has to follow the structure of a PDB with each PDB data block mapped to a NeXus group of class NXpdb, using a lowercase version of the data block name as the name of the NeXus group, each PDB category in that data block mapped to a NeXus group of class NXpdb and with each PDB column mapped to a NeXus field. Each column in a looped PDB category should always be presented as a 1-dimensional array. The columns in an unlooped PDB category should be presented as scalar values. If a PDB category specifies particular units for columns, the same units should be used for the corresponding fields.

A PDB entry is unambiguous when all information is carried as text. All text data should be presented as quoted strings, with the quote marks except for the null values “.” or “?”

For clarity in NXpdb form, numeric data may be presented using the numeric types specified in the mmCIF dictionary. In that case, if a PDB null value, “.” or “?”, is contained in a numeric column, the IEEE nan should be used for “?” and the IEEE inf should be used for “.”.

An arbitrary DDL2 CIF file can be represented in NeXus using NXpdb. However, if save frames are required, an NXpdb_class attribute with the value “CBF_cbfds” is required for each NeXus group representing a save frame. NXpdb attributes are not required for other CIF components, but may be used to provide internal documentation.

The nesting of NXpdb groups and datasets that correspond to a CIF with two categories and one saveframe, including the NXpdb_class attributes is:

```
(datablock1):NXpdb
@NXpdb_class:CBF_cbfds
(category1):NXpdb
@NXpdb_class:CBF_cbfcat
  (column_name1):[...]
  (column_name2):[...]
  (column_name3):[...]
...
(category2):NXpdb
```
For example, a PDB entry that begins:

```
data_1YVA
  #
  _entry.id 1YVA
  #
  _audit_conform.dict_name mmcif_pdbx.dic
  _audit_conform.dict_version 5.279
  _audit_conform.dict_location http://mmcif.pdb.org/dictionaries/ascii/mmCIF_pdbx.dic
  #
  loop_
  _database_2.database_id
  _database_2.database_code
  PDB 1YVA
  RCSB RCSB031959
  WWPDB D_1000031959
  #
```

would produce:

```
sample:NXSampLe
  1yva:NXpdb
  entry:NXpdb
    id:"1YVA"
  audit_conform:NXpdb
    dict_name:"mmCIF_pdbx.dic"
    dict_version:"5.279"
    dict_location:"http://mmcif.pdb.org/dictionaries/ascii/mmCIF_pdbx.dic"
  database_2:NXpdb
    database_id:["PDB","RCSB","WWPDB"]
    database_code:["1YVA","RCSB031959","D_1000031959"]
```

another example is the following excerpt from pdb entry 9ins, giving the sequences of the two chains:

```
loop_
  _entity_poly.entity_id
```
which converts to:

```plaintext
entity_poly:NXpdb
  @NXpdb_class:CBF_cbfcat
  entity_id:["1", "2"]
  nstd_linkage:["no", "no"]
  nstd_monomer:["no", "no"]
  pdbx_seq_one_letter_code:["GIVEQCCTSICSLYQLENYCN",
  → "FVNQHLCGSHLVEALYLVCGERGFFYTPKA"]
  pdbx_seq_one_letter_code_can:["GIVEQCCTSICSLYQLENYCN",
  → "FVNQHLCGSHLVEALYLVCGERGFFYTPKA"]
  type:["polypeptide(L)", "polypeptide(L)"
```

Symbols:

- No symbol table

Groups cited:

- none

Structure:

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/base_classes/NXpdb.nxdl.xml

**NXpinhole**

Status:

- base class, extends **NXobject**

Description:

A simple pinhole.

For more complex geometries, **NXaperture** should be used.

Symbols:

- No symbol table

Groups cited:

- **NXtransformations**

Structure:

- `@default`: (optional) **NX_CHAR**

  Declares which child group contains a path leading to a **NXdata** group.
It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

**depends_on:** (optional) *NX CHAR*

Points to the path of the last element in the geometry chain that places this object in space. When followed through that chain is supposed to end at an element depending on “.” i.e. the origin of the coordinate system. If desired the location of the slit can also be described relative to an NXbeam, which will allow a simple description of a non-centred pinhole.

The reference direction of the pinhole is parallel with the z axis. The reference point of the pinhole is its center in the x and y axis. The reference point on the z axis is the plane which overlaps the side of the opening of the pin hole pointing towards the source (minus on the z axis).

**diameter:** (optional) *NX NUMBER* \{units=*NX LENGTH*\}

Size of the circular hole defining the transmitted beam size.

**TRANSFORMATIONS:** (optional) *NX transformations*

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXpinhole/depends_on-field
- /NXpinhole/diameter-field
- /NXpinhole/TRANSFORMATIONS-group
- /NXpinhole@default-attribute

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/base_classes/NXpinhole.nxdl.xml

**NXpolarizer**

**Status:**

base class, extends *NXobject*

**Description:**

A spin polarizer.

**Symbols:**

No symbol table
Groups cited:

NXtransformations

Structure:

@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

type: (optional) NX_CHAR

one of these values: “crystal”, “supermirror”, “3He”

composition: (optional) NX_CHAR

description of the composition of the polarizing material

reflection: (optional) NX_INT (Rank: 1, Dimensions: [3]) {units=NX_UNITLESS}

[hkl] values of nominal reflection

efficiency: (optional) NX_FLOAT {units=NX_DIMENSIONLESS}

polarizing efficiency

depends_on: (optional) NX_CHAR

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

TRANSFORMATIONS: (optional) NXtransformations

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXpolarizer/composition-field
- /NXpolarizer/depends_on-field
- /NXpolarizer/efficiency-field
- /NXpolarizer/reflection-field
- /NXpolarizer/TRANSFORMATIONS-group
- /NXpolarizer/type-field
- /NXpolarizer@default-attribute

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/base_classes/NXpolarizer.nxdl.xml
**NXpositioner**

**Status:**
base class, extends *NXObject*

**Description:**
A generic positioner such as a motor or piezo-electric transducer.

**Symbols:**
No symbol table

**Groups cited:**
*NXtransformations*

**Structure:**

@default: (optional) *NX_CHAR*
Declares which child group contains a path leading to a *NXdata* group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See [https://www.nexusformat.org/2014_How_to_find_default_data.html](https://www.nexusformat.org/2014_How_to_find_default_data.html) for a summary of the discussion.

name: (optional) *NX_CHAR*
symbolic or mnemonic name (one word)

description: (optional) *NX_CHAR*
description of positioner

value: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n]) {units=*NX_ANY*}
best known value of positioner - need [n] as may be scanned

raw_value: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n]) {units=*NX_ANY*}
raw value of positioner - need [n] as may be scanned

target_value: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n]) {units=*NX_ANY*}
targeted (commanded) value of positioner - need [n] as may be scanned

tolerance: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n]) {units=*NX_ANY*}
maximum allowable difference between target_value and value

soft_limit_min: (optional) *NX_NUMBER* {units=*NX_ANY*}
minimum allowed limit to set value

soft_limit_max: (optional) *NX_NUMBER* {units=*NX_ANY*}
maximum allowed limit to set value

velocity: (optional) *NX_NUMBER* {units=*NX_ANY*}
velocity of the positioner (distance moved per unit time)

acceleration_time: (optional) *NX_NUMBER* {units=*NX_ANY*}
time to ramp the velocity up to full speed

controller_record: (optional) *NX_CHAR*
Hardware device record, e.g. EPICS process variable, taco/tango …

3.3. NeXus Class Definitions 273
depends_on: (optional) `NX_CHAR`

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

**TRANSFORMATIONS**: (optional) `NXtransformations`

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXpositioner/acceleration_time-field`
- `/NXpositioner/controller_record-field`
- `/NXpositioner/depends_on-field`
- `/NXpositioner/description-field`
- `/NXpositioner/name-field`
- `/NXpositioner/raw_value-field`
- `/NXpositioner/soft_limit_max-field`
- `/NXpositioner/soft_limit_min-field`
- `/NXpositioner/target_value-field`
- `/NXpositioner/tolerance-field`
- `/NXpositioner/TRANSFORMATIONS-group`
- `/NXpositioner/value-field`
- `/NXpositioner/velocity-field`
- `/NXpositioner@default-attribute`

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/base_classes/NXpositioner.nxdl.xml

**NXprocess**

**Status:**

base class, extends `NXobject`

**Description:**

Document an event of data processing, reconstruction, or analysis for this data.

**Symbols:**

No symbol table
Groups cited:
  NXnote

Structure:

@default: (optional) NX_CHAR
  Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

program: (optional) NX_CHAR
  Name of the program used

sequence_index: (optional) NX_POSINT
  Sequence index of processing, for determining the order of multiple NXprocess steps. Starts with 1.

version: (optional) NX_CHAR
  Version of the program used

date: (optional) NX_DATE_TIME
  Date and time of processing.

NOTE: (optional) NXnote
  The note will contain information about how the data was processed or anything about the data provenance. 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.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXprocess/date-field
- /NXprocess/NOTE-group
- /NXprocess/program-field
- /NXprocess/sequence_index-field
- /NXprocess/version-field
- /NXprocess@default-attribute

NXDL Source:
  https://github.com/nexusformat/definitions/blob/main/base_classes/NXprocess.nxdl.xml
NXreflections

Status:

base class, extends NXobject

Description:

Reflection data from diffraction experiments

Symbols:

n: number of reflections
m: number of experiments

Groups cited:

none

Structure:

@description: (optional) NX_CHAR

Describes the dataset

@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

experiments: (optional) NX_CHAR (Rank: 1, Dimensions: [m])

The experiments from which the reflection data derives

h: (optional) NX_NUMBER (Rank: 1, Dimensions: [n])

The h component of the miller index

@description: (optional) NX_CHAR

Describes the dataset

k: (optional) NX_NUMBER (Rank: 1, Dimensions: [n])

The k component of the miller index

@description: (optional) NX_CHAR

Describes the dataset

l: (optional) NX_NUMBER (Rank: 1, Dimensions: [n])

The l component of the miller index

@description: (optional) NX_CHAR

Describes the dataset

id: (optional) NX_INT (Rank: 1, Dimensions: [n])

The id of the experiment which resulted in the reflection. If the value is greater than 0, the experiments must link to a multi-experiment NXmx group

@description: (optional) NX_CHAR

Describes the dataset
**reflection_id**: (optional) *NX_INT* (Rank: 1, Dimensions: \([n]\))

The id of the reflection. Multiple partials from the same reflection should all have the same id

@**description**: (optional) *NX_CHAR*

Describes the dataset

**entering**: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: \([n]\))

Is the reflection entering or exiting the Ewald sphere

@**description**: (optional) *NX_CHAR*

Describes the dataset

**det_module**: (optional) *NX_INT* (Rank: 1, Dimensions: \([n]\))

The detector module on which the reflection was recorded

@**description**: (optional) *NX_CHAR*

Describes the dataset

**flags**: (optional) *NX_INT* (Rank: 1, Dimensions: \([n]\))

Status flags describing the reflection.

This is a bit mask. The bits in the mask follow the convention used by DIALS, and have the following names:

<table>
<thead>
<tr>
<th>bit</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>predicted</td>
</tr>
<tr>
<td>1</td>
<td>observed</td>
</tr>
<tr>
<td>2</td>
<td>indexed</td>
</tr>
<tr>
<td>3</td>
<td>used_in_refinement</td>
</tr>
<tr>
<td>4</td>
<td>strong</td>
</tr>
<tr>
<td>5</td>
<td>reference_spot</td>
</tr>
<tr>
<td>6</td>
<td>dont_integrate</td>
</tr>
<tr>
<td>7</td>
<td>integrated_sum</td>
</tr>
<tr>
<td>8</td>
<td>integrated_prf</td>
</tr>
<tr>
<td>9</td>
<td>integrated</td>
</tr>
<tr>
<td>10</td>
<td>overloaded</td>
</tr>
<tr>
<td>11</td>
<td>overlapped</td>
</tr>
<tr>
<td>12</td>
<td>overlapped_fg</td>
</tr>
<tr>
<td>13</td>
<td>in_powder_ring</td>
</tr>
<tr>
<td>14</td>
<td>foreground_includes_bad_pixels</td>
</tr>
<tr>
<td>15</td>
<td>background_includes_bad_pixels</td>
</tr>
<tr>
<td>16</td>
<td>includes_bad_pixels</td>
</tr>
<tr>
<td>17</td>
<td>bad_shoebox</td>
</tr>
<tr>
<td>18</td>
<td>bad_spot</td>
</tr>
<tr>
<td>19</td>
<td>used_in_modelling</td>
</tr>
<tr>
<td>20</td>
<td>centroid_outlier</td>
</tr>
<tr>
<td>21</td>
<td>failed_during_background_modelling</td>
</tr>
<tr>
<td>22</td>
<td>failed_during_summation</td>
</tr>
<tr>
<td>23</td>
<td>failed_during_profile_fitting</td>
</tr>
<tr>
<td>24</td>
<td>bad_reference</td>
</tr>
</tbody>
</table>

@**description**: (optional) *NX_CHAR*
Describes the dataset

d: (optional) NX_FLOAT (Rank: 1, Dimensions: [n])
The resolution of the reflection
@description: (optional) NX_CHAR
  Describes the dataset

partiality: (optional) NX_FLOAT (Rank: 1, Dimensions: [n])
The partiality of the reflection. Dividing by this number will inflate the measured intensity to
the full reflection equivalent.
@description: (optional) NX_CHAR
  Describes the dataset

predicted_frame: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_UNITLESS}
The frame on which the bragg peak of the reflection is predicted
@description: (optional) NX_CHAR
  Describes the dataset

predicted_x: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_LENGTH}
The x position at which the bragg peak of the reflection is predicted
@description: (optional) NX_CHAR
  Describes the dataset

predicted_y: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_LENGTH}
The y position at which the bragg peak of the reflection is predicted
@description: (optional) NX_CHAR
  Describes the dataset

predicted_phi: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_ANGLE}
The phi angle at which the bragg peak of the reflection is predicted
@description: (optional) NX_CHAR
  Describes the dataset

predicted_px_x: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_UNITLESS}
The x pixel position at which the bragg peak of the reflection is predicted
@description: (optional) NX_CHAR
  Describes the dataset

predicted_px_y: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_UNITLESS}
The y pixel position at which the bragg peak of the reflection is predicted
@description: (optional) NX_CHAR
  Describes the dataset

observed_frame: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_UNITLESS}
The estimate of the frame at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

observed_frame_var: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_UNITLESS}

The variance on the estimate of the frame at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

observed_frame_errors: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_UNITLESS}

The standard deviation of the estimate of the frame at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

observed_px_x: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_UNITLESS}

The estimate of the pixel x position at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

observed_px_x_var: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_UNITLESS}

The variance on the estimate of the pixel x position at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

observed_px_x_errors: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_UNITLESS}

The standard deviation of the estimate of the pixel x position at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

observed_px_y: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_UNITLESS}

The estimate of the pixel y position at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

observed_px_y_var: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_UNITLESS}

The variance on the estimate of the pixel y position at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

observed_px_y_errors: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_UNITLESS}
The standard deviation of the estimate of the pixel y position at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

**observed_phi**: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_ANGLE}

The estimate of the phi angle at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

**observed_phi_var**: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_ANGLE}

The variance on the estimate of the phi angle at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

**observed_phi_errors**: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_ANGLE}

The standard deviation of the estimate of the phi angle at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

**observed_x**: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_LENGTH}

The estimate of the x position at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

**observed_x_var**: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_LENGTH}

The variance on the estimate of the x position at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

**observed_x_errors**: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_LENGTH}

The standard deviation of the estimate of the x position at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

**observed_y**: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_LENGTH}

The estimate of the y position at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset

**observed_y_var**: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_LENGTH}

The standard deviation of the estimate of the y position at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR

Describes the dataset
The variance on the estimate of the y position at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR
Describes the dataset

observed_y_errors: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_LENGTH}
The standard deviation of the estimate of the y position at which the central impact of the reflection was recorded

@description: (optional) NX_CHAR
Describes the dataset

bounding_box: (optional) NX_INT (Rank: 2, Dimensions: [n, 6]) {units=NX_UNITLESS}
The bounding box around the recorded reflection. Should be an integer array of length 6, where the 6 values are pixel positions or frame numbers, as follows:

<table>
<thead>
<tr>
<th>Index</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The lower pixel x position</td>
</tr>
<tr>
<td>1</td>
<td>The upper pixel x position</td>
</tr>
<tr>
<td>2</td>
<td>The lower pixel y position</td>
</tr>
<tr>
<td>3</td>
<td>The upper pixel y position</td>
</tr>
<tr>
<td>4</td>
<td>The lower frame number</td>
</tr>
<tr>
<td>5</td>
<td>The upper frame number</td>
</tr>
</tbody>
</table>

@description: (optional) NX_CHAR
Describes the dataset

background_mean: (optional) NX_FLOAT (Rank: 1, Dimensions: [n])
The mean background under the reflection peak

@description: (optional) NX_CHAR
Describes the dataset

int_prf: (optional) NX_FLOAT (Rank: 1, Dimensions: [n])
The estimate of the reflection intensity by profile fitting

@description: (optional) NX_CHAR
Describes the dataset

int_prf_var: (optional) NX_FLOAT (Rank: 1, Dimensions: [n])
The variance on the estimate of the reflection intensity by profile fitting

@description: (optional) NX_CHAR
Describes the dataset

int_prf_errors: (optional) NX_FLOAT (Rank: 1, Dimensions: [n])
The standard deviation of the estimate of the reflection intensity by profile fitting

@description: (optional) NX_CHAR
Describes the dataset

int_sum: (optional) NX_FLOAT (Rank: 1, Dimensions: [n])
The estimate of the reflection intensity by summation

@description: (optional) NX_CHAR

Describes the dataset

int_sum_var: (optional) NX_FLOAT (Rank: 1, Dimensions: [n])

The variance on the estimate of the reflection intensity by summation

@description: (optional) NX_CHAR

Describes the dataset

int_sum_errors: (optional) NX_FLOAT (Rank: 1, Dimensions: [n])

The standard deviation of the estimate of the reflection intensity by summation

@description: (optional) NX_CHAR

Describes the dataset

lp: (optional) NX_FLOAT (Rank: 1, Dimensions: [n])

The LP correction factor to be applied to the reflection intensities

@description: (optional) NX_CHAR

Describes the dataset

prf_cc: (optional) NX_FLOAT (Rank: 1, Dimensions: [n])

The correlation of the reflection profile with the reference profile used in profile fitting

@description: (optional) NX_CHAR

Describes the dataset

overlaps: (optional) NX_INT

An adjacency list specifying the spatial overlaps of reflections. The adjacency list is specified using an array data type where the elements of the array are the indices of the adjacent overlapped reflection

@description: (optional) NX_CHAR

Describes the dataset

polar_angle: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_ANGLE}

Polar angle of reflection centroid, following the NeXus simple (spherical polar) coordinate system

@description: (optional) NX_CHAR

Describes the dataset

azimuthal_angle: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_ANGLE}

Azimuthal angle of reflection centroid, following the NeXus simple (spherical polar) coordinate system
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXreflections/azimuthal_angle-field
- /NXreflections/background_mean-field
- /NXreflections/background_mean@description-attribute
- /NXreflections/bounding_box-field
- /NXreflections/bounding_box@description-attribute
- /NXreflections/d-field
- /NXreflections/d@description-attribute
- /NXreflections/det_module-field
- /NXreflections/det_module@description-attribute
- /NXreflections/entering-field
- /NXreflections/entering@description-attribute
- /NXreflections/experiments-field
- /NXreflections/flags-field
- /NXreflections/flags@description-attribute
- /NXreflections/h-field
- /NXreflections/h@description-attribute
- /NXreflections/id-field
- /NXreflections/id@description-attribute
- /NXreflections/int_prf-field
- /NXreflections/int_prf@description-attribute
- /NXreflections/int_prf_errors-field
- /NXreflections/int_prf_errors@description-attribute
- /NXreflections/int_prf_var-field
- /NXreflections/int_prf_var@description-attribute
- /NXreflections/int_sum-field
- /NXreflections/int_sum@description-attribute
- /NXreflections/int_sum_errors-field
- /NXreflections/int_sum_errors@description-attribute
- /NXreflections/int_sum_var-field
- /NXreflections/int_sum_var@description-attribute
- /NXreflections/k-field
- /NXreflections/k@description-attribute
- /NXreflections/l-field
- /NXreflections/l@description-attribute
- /NXreflections/lp-field
- /NXreflections/lp@description-attribute
- /NXreflections/observed_frame-field
- /NXreflections/observed_frame@description-attribute
- /NXreflections/observed_frame_errors-field
- /NXreflections/observed_frame_errors@description-attribute
- /NXreflections/observed_frame_var-field
- /NXreflections/observed_frame_var@description-attribute
- /NXreflections/observed_phi-field
- /NXreflections/observed_phi@description-attribute
- /NXreflections/observed_phi_errors-field
- /NXreflections/observed_phi_errors@description-attribute
- /NXreflections/observed_phi_var-field
- /NXreflections/observed_phi_var@description-attribute
- /NXreflections/observed_px_x-field
- /NXreflections/observed_px_x@description-attribute
- /NXreflections/observed_px_x_errors-field
- /NXreflections/observed_px_x_errors@description-attribute
- /NXreflections/observed_px_x_var-field
- /NXreflections/observed_px_x_var@description-attribute
- /NXreflections/observed_px_y-field
- /NXreflections/observed_px_y@description-attribute
- /NXreflections/observed_px_y_errors-field
- /NXreflections/observed_px_y_errors@description-attribute
- /NXreflections/observed_px_y_var-field
- /NXreflections/observed_px_y_var@description-attribute
- /NXreflections/observed_x-field
- /NXreflections/observed_x@description-attribute
- /NXreflections/observed_x_errors-field
- /NXreflections/observed_x_errors@description-attribute
- /NXreflections/observed_x_var-field
- /NXreflections/observed_x_var@description-attribute
- /NXreflections/observed_y-field
- /NXreflections/observed_y@description-attribute
- /NXreflections/observed_y_errors-field
- /NXreflections/observed_y_errors@description-attribute
• /NXreflections/observed_y_var-field
• /NXreflections/observed_y_var@description-attribute
• /NXreflections/overlaps-field
• /NXreflections/overlaps@description-attribute
• /NXreflections/partiality-field
• /NXreflections/partiality@description-attribute
• /NXreflections/polar_angle-field
• /NXreflections/polar_angle@description-attribute
• /NXreflections/predicted_frame-field
• /NXreflections/predicted_frame@description-attribute
• /NXreflections/predicted_phi-field
• /NXreflections/predicted_phi@description-attribute
• /NXreflections/predicted_px_x-field
• /NXreflections/predicted_px_x@description-attribute
• /NXreflections/predicted_px_y-field
• /NXreflections/predicted_px_y@description-attribute
• /NXreflections/predicted_x-field
• /NXreflections/predicted_x@description-attribute
• /NXreflections/predicted_y-field
• /NXreflections/predicted_y@description-attribute
• /NXreflections/prf_cc-field
• /NXreflections/prf_cc@description-attribute
• /NXreflections/reflection_id-field
• /NXreflections/reflection_id@description-attribute
• /NXreflections@default-attribute
• /NXreflections@description-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXreflections.nxdl.xml

**NXroot**

**Status:**

base class, extends *NXobject*

**Description:**

Definition of the root NeXus group.

**Symbols:**

No symbol table
Groups cited:

NXentry

Structure:

@NX_class: (optional) NX_CHAR

The root of any NeXus data file is an NXroot class (no other choice is allowed for a valid NeXus data file). This attribute cements that definition.

Obligatory value: NXroot

@file_time: (optional) NX_DATE_TIME

Date and time file was originally created

@file_name: (optional) NX_CHAR

File name of original NeXus file

@file_update_time: (optional) NX_DATE_TIME

Date and time of last file change at close

@NeXus_version: (optional) NX_CHAR

Version of NeXus API used in writing the file.

Only used when the NAPI has written the file. Note that this is different from the version of the base class or application definition version number.

@HDF_version: (optional) NX_CHAR

Version of HDF (version 4) library used in writing the file

@HDF5_Version: (optional) NX_CHAR

Version of HDF5 library used in writing the file.

Note this attribute is spelled with uppercase “V”, different than other version attributes.

@XML_version: (optional) NX_CHAR

Version of XML support library used in writing the XML file

@h5py_version: (optional) NX_CHAR

Version of h5py Python package used in writing the file

@creator: (optional) NX_CHAR

facility or program where file originated

@creator_version: (optional) NX_CHAR

Version of facility or program used in writing the file

@default: (optional) NX_CHAR

Declares which NXentry group contains the data to be shown by default. It is used to resolve ambiguity when more than one NXentry group exists. The value names the default NXentry group. The value must be the name of a child of the current group. The child must be a NeXus group or a link to a NeXus group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

ENTRY: (optional) NXentry
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXroot/ENTRY-group`
- `/NXroot@creator-attribute`
- `/NXroot@creator_version-attribute`
- `/NXroot@default-attribute`
- `/NXroot@file_name-attribute`
- `/NXroot@file_time-attribute`
- `/NXroot@file_update_time-attribute`
- `/NXroot@h5py_version-attribute`
- `/NXroot@HDF5_Version-attribute`
- `/NXroot@HDF_version-attribute`
- `/NXroot@NeXus_version-attribute`
- `/NXroot@NX_class-attribute`
- `/NXroot@XML_version-attribute`

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXroot.nxdl.xml

NXSample

Status:
base class, extends `NXobject`

Description:
Any information on 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.

Symbols:
symbolic array lengths to be coordinated between various fields

- `n_comp`: number of compositions
- `n_Temp`: number of temperatures
- `n_eField`: number of values in applied electric field
- `n_mField`: number of values in applied magnetic field
- `n_pField`: number of values in applied pressure field
- `n_sField`: number of values in applied stress field
Groups cited:
NXbeam, NXdata, NXenvironment, NXgeometry, NXlog, NXoff_geometry, NXpositioner, NXsample_component, NXtransformations

Structure:

@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

name: (optional) NX_CHAR

Descriptive name of sample

chemical_formula: (optional) 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: (optional) NX_FLOAT (Rank: anyRank, Dimensions: [n.Temp]) {units=NX_TEMPERATURE}

Sample temperature. This could be a scanned variable

electric_field: (optional) NX_FLOAT (Rank: 1, Dimensions: [n_eField]) {units=NX_VOLTAGE}

Applied electric field

@direction: (optional) NX_CHAR

Any of these values: x | y | z

magnetic_field: (optional) NX_FLOAT (Rank: 1, Dimensions: [n_mField]) {units=NX_ANY}

Applied magnetic field

@direction: (optional) NX_CHAR

Any of these values: x | y | z

stress_field: (optional) NX_FLOAT (Rank: 1, Dimensions: [n_sField]) {units=NX_ANY}
Applied external stress field

@direction: (optional) NX_CHAR

Any of these values: x | y | z

pressure: (optional) NX_FLOAT (Rank: 1, Dimensions: [n_pField]) {units=NX_PRESSURE}

Applied pressure

changer_position: (optional) NX_INT {units=NX_UNITLESS}

Sample changer position

unit_cell_abc: (optional) NX_FLOAT (Rank: 1, Dimensions: [3]) {units=NX_LENGTH}

Crystallography unit cell parameters a, b, and c

unit_cell_alphabetagamma: (optional) NX_FLOAT (Rank: 1, Dimensions: [3]) {units=NX_ANGLE}

Crystallography unit cell parameters alpha, beta, and gamma

unit_cell: (optional) NX_FLOAT (Rank: 2, Dimensions: [n_comp, 6]) {units=NX_LENGTH}

Unit cell parameters (lengths and angles)

unit_cell_volume: (optional) NX_FLOAT (Rank: 1, Dimensions: [n_comp]) {units=NX_VOLUME}

Volume of the unit cell

sample_orientation: (optional) NX_FLOAT (Rank: 1, Dimensions: [3]) {units=NX_ANGLE}


orientation_matrix: (optional) NX_FLOAT (Rank: 3, Dimensions: [n_comp, 3, 3])


ub_matrix: (optional) NX_FLOAT (Rank: 3, Dimensions: [n_comp, 3, 3])

UB matrix of single crystal sample using Busing-Levy convention: W. R. Busing and H. A. Levy (1967). Acta Cryst. 22, 457-464. This is the multiplication of the orientation_matrix, given above, with the \( B \) matrix which can be derived from the lattice constants.

mass: (optional) NX_FLOAT (Rank: 1, Dimensions: [n_comp]) {units=NX_MASS}

Mass of sample

density: (optional) NX_FLOAT (Rank: 1, Dimensions: [n_comp]) {units=NX_MASS_DENSITY}

Density of sample

relative_molecular_mass: (optional) NX_FLOAT (Rank: 1, Dimensions: [n_comp]) {units=NX_MASS}

Relative Molecular Mass of sample

type: (optional) NX_CHAR

Any of these values:

• sample
• sample+can
• can
• sample+buffer
- buffer
- calibration sample
- normalisation sample
- simulated data
- none
- sample environment

**situation:** (optional) **NX_CHAR**

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.

Any of these values:
- air
- vacuum
- inert atmosphere
- oxidising atmosphere
- reducing atmosphere
- sealed can
- other

**description:** (optional) **NX_CHAR**

Description of the sample

**preparation_date:** (optional) **NX_DATE_TIME**

Date of preparation of the sample

**component:** (optional) **NX_CHAR** (Rank: 1, Dimensions: [n_comp])

Details of the component of the sample and/or can

**sample_component:** (optional) **NX_CHAR** (Rank: 1, Dimensions: [n_comp])

Type of component

Any of these values: sample | can | atmosphere | kit

**concentration:** (optional) **NX_FLOAT** (Rank: 1, Dimensions: [n_comp]) {units=NX_MASS_DENSITY}

Concentration of each component

**volume_fraction:** (optional) **NX_FLOAT** (Rank: 1, Dimensions: [n_comp])

Volume fraction of each component

**scattering_length_density:** (optional) **NX_FLOAT** (Rank: 1, Dimensions: [n_comp]) {units=NX_SCATTERING_LENGTH_DENSITY}

Scattering length density of each component

**unit_cell_class:** (optional) **NX_CHAR**

In case it is all we know and we want to record/document it

Any of these values:
- triclinic
• monoclinic
• orthorhombic
• tetragonal
• rhombohedral
• hexagonal
• cubic

**space_group**: (optional) `NX_CHAR` (Rank: 1, Dimensions: [n_comp])
Crystallographic space group

**point_group**: (optional) `NX_CHAR` (Rank: 1, Dimensions: [n_comp])
Crystallographic point group, deprecated if `space_group` present

**path_length**: (optional) `NX_FLOAT` {units=`NX_LENGTH` }
Path length through sample/can for simple case when it does not vary with scattering direction

**path_length_window**: (optional) `NX_FLOAT` {units=`NX_LENGTH` }
Thickness of a beam entry/exit window on the can (mm) - assumed same for entry and exit

**thickness**: (optional) `NX_FLOAT` {units=`NX_LENGTH` }
sample thickness

**external_DAC**: (optional) `NX_FLOAT` {units=`NX_ANY` }
value sent to user’s sample setup

**short_title**: (optional) `NX_CHAR`
20 character fixed length sample description for legends

**rotation_angle**: (optional) `NX_FLOAT` {units=`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. Note, it is recommended to use NXtransformations instead.

**x_translation**: (optional) `NX_FLOAT` {units=`NX_LENGTH` }
Translation of the sample along the X-direction of the laboratory coordinate system Note, it is recommended to use NXtransformations instead.

**distance**: (optional) `NX_FLOAT` {units=`NX_LENGTH` }
Translation of the sample along the Z-direction of the laboratory coordinate system. Note, it is recommended to use NXtransformations instead.

**depends_on**: (optional) `NX_CHAR`
NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

**geometry**: (optional) `NXgeometry`
**DEPRECATED**: Use the field `depends_on` and `NXtransformations` to position the sample and `NXoff_geometry` to describe its shape instead.

The position and orientation of the center of mass of the sample

**BEAM**: (optional) `NXbeam`

Details of beam incident on sample - used to calculate sample/beam interaction point

**SAMPLE_COMPONENT**: (optional) `NXsample_component`

One group per sample component. This is the preferred way of recording per component information over the n_comp arrays.

**transmission**: (optional) `NXdata`

As a function of Wavelength

**temperature_log**: (optional) `NXlog`

**DEPRECATED**: use `temperature`, see: https://github.com/nexusformat/definitions/issues/816

temperature_log.value is a link to e.g. temperature_env.sensor1.value_log.value

**temperature_env**: (optional) `NXenvironment`

Additional sample temperature environment information

**magnetic_field**: (optional) `NXlog`

magnetic_field.value is a link to e.g. magnetic_field_env.sensor1.value

**magnetic_field_log**: (optional) `NXlog`

**DEPRECATED**: use `magnetic_field`, see: https://github.com/nexusformat/definitions/issues/816

magnetic_field_log.value is a link to e.g. magnetic_field_env.sensor1.value_log.value

**magnetic_field_env**: (optional) `NXenvironment`

Additional sample magnetic environment information

**external_ADC**: (optional) `NXlog`

logged value (or logic state) read from user’s setup

**POSITIONER**: (optional) `NXpositioner`

Any positioner (motor, PZT, ...) used to locate the sample

**OFF_GEOMETRY**: (optional) `NXoff_geometry`

This group describes the shape of the sample

**TRANSFORMATIONS**: (optional) `NXtransformations`

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.
**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXsample/BEAM-group`
- `/NXsample/changer_position-field`
- `/NXsample/chemical_formula-field`
- `/NXsample/component-field`
- `/NXsample/concentration-field`
- `/NXsample/density-field`
- `/NXsample/depends_on-field`
- `/NXsample/description-field`
- `/NXsample/distance-field`
- `/NXsample/electric_field-field`
- `/NXsample/electric_field@direction-attribute`
- `/NXsample/external_ADC-group`
- `/NXsample/external_DAC-field`
- `/NXsample/geometry-group`
- `/NXsample/magnetic_field-field`
- `/NXsample/magnetic_field-group`
- `/NXsample/magnetic_field@direction-attribute`
- `/NXsample/magnetic_field_env-group`
- `/NXsample/magnetic_field_log-group`
- `/NXsample/mass-field`
- `/NXsample/name-field`
- `/NXsample/OFF_GEOMETRY-group`
- `/NXsample/orientation_matrix-field`
- `/NXsample/path_length-field`
- `/NXsample/path_length_window-field`
- `/NXsample/point_group-field`
- `/NXsample/POSITIONER-group`
- `/NXsample/preparation_date-field`
- `/NXsample/pressure-field`
- `/NXsample/relative_molecular_mass-field`
- `/NXsample/rotation_angle-field`
- `/NXsample/sample_component-field`
- `/NXsample/SAMPLE_COMPONENT-group`
- `/NXsample/sample_orientation-field`
NXsample_component

Status:

base class, extends NXobject

Description:

One group like this per component can be recorded for a sample consisting of multiple components.

Symbols:

symbolic array lengths to be coordinated between various fields

n_Temp: number of temperatures

n_eField: number of values in applied electric field

n_mField: number of values in applied magnetic field

n_pField: number of values in applied pressure field
**n_sField**: number of values in applied stress field

**Groups cited:**
- **NXdata**

**Structure:**

- **@default**: (optional) **NX_CHAR**
  
  Declares which child group contains a path leading to a **NXdata** group.

  It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See [https://www.nexusformat.org/2014_How_to_find_default_data.html](https://www.nexusformat.org/2014_How_to_find_default_data.html) for a summary of the discussion.

- **name**: (optional) **NX_CHAR**
  
  Descriptive name of sample component

- **chemical_formula**: (optional) **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.

- **unit_cell_abc**: (optional) **NX_FLOAT** (Rank: 1, Dimensions: [3]) `{units=NX_LENGTH}`
  
  Crystallography unit cell parameters a, b, and c

- **unit_cell_alphabetagamma**: (optional) **NX_FLOAT** (Rank: 1, Dimensions: [3]) `{units=NX_ANGLE}`
  
  Crystallography unit cell parameters alpha, beta, and gamma

- **unit_cell_volume**: (optional) **NX_FLOAT** `{units=NX_VOLUME}`
  
  Volume of the unit cell

- **sample_orientation**: (optional) **NX_FLOAT** (Rank: 1, Dimensions: [3]) `{units=NX_ANGLE}`
  
  This will follow the Busing and Levy convention from Acta.Crysta v22, p457 (1967)

- **orientation_matrix**: (optional) **NX_FLOAT** (Rank: 2, Dimensions: [3, 3])
  
  Orientation matrix of single crystal sample component. This will follow the Busing and Levy convention from Acta.Crysta v22, p457 (1967)

- **mass**: (optional) **NX_FLOAT** `{units=NX_MASS}`

3.3. NeXus Class Definitions 295
Mass of sample component

**density**: (optional) `NX_FLOAT` `{units=NX_MASS_DENSITY}`

Density of sample component

**relative_molecular_mass**: (optional) `NX_FLOAT` `{units=NX_MASS}`

Relative Molecular Mass of sample component

**description**: (optional) `NX_CHAR`

Description of the sample component

**volume_fraction**: (optional) `NX_FLOAT`

Volume fraction of component

**scattering_length_density**: (optional) `NX_FLOAT` `{units=NX_SCATTERING_LENGTH_DENSITY}`

Scattering length density of component

**unit_cell_class**: (optional) `NX_CHAR`

In case it is all we know and we want to record/document it

Any of these values:

- triclinic
- monoclinic
- orthorhombic
- tetragonal
- rhombohedral
- hexagonal
- cubic

**space_group**: (optional) `NX_CHAR`

Crystallographic space group

**point_group**: (optional) `NX_CHAR`

Crystallographic point group, deprecated if space_group present

**transmission**: (optional) `NXdata`

As a function of Wavelength

---

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXsample_component/chemical_formula-field`
- `/NXsample_component/density-field`
- `/NXsample_component/description-field`
- `/NXsample_component/mass-field`
- `/NXsample_component/name-field`
- `/NXsample_component/orientation_matrix-field`
NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXsample_component.nxdl.xml

NXsensor

Status:
base class, extends NXobject

Description:
A sensor used to monitor an external condition
The condition itself is described in NXenvironment.

Symbols:
No symbol table

Groups cited:
NXgeometry, NXlog, NXoff_geometry, NXorientation, NXtransformations

Structure:

@default: (optional) NX_CHAR
Declares which child group contains a path leading to a NXdata group.
It is recommended (as of NIAC2014) to use this attribute to help define the path to the default
dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html
for a summary of the discussion.

model: (optional) NX_CHAR
Sensor identification code/model number

name: (optional) NX_CHAR
Name for the sensor

short_name: (optional) NX_CHAR
Short name of sensor used e.g. on monitor display program
attached_to: (optional) NX_CHAR

where sensor is attached to ("sample" | "can")

measurement: (optional) NX_CHAR

name for measured signal

Any of these values:
  • temperature
  • pH
  • magnetic_field
  • electric_field
  • conductivity
  • resistance
  • voltage
  • pressure
  • flow
  • stress
  • strain
  • shear
  • surface_pressure

type: (optional) 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: (optional) NX_BOOLEAN

Is data collection controlled or synchronised to this quantity: 1=no, 0=to “value”, 1=to “value_deriv1”, etc.

high_trip_value: (optional) NX_FLOAT {units=NX_ANY}

Upper control bound of sensor reading if using run_control

low_trip_value: (optional) NX_FLOAT {units=NX_ANY}

Lower control bound of sensor reading if using run_control

value: (optional) NX_FLOAT (Rank: 1, Dimensions: [n]) {units=NX_ANY}
nominal setpoint or average value - need [n] as may be a vector

**value_deriv1**: (optional) *NX_FLOAT* (Rank: same as field value, Dimensions: same as field value)
{units=*NX_ANY*}

Nominal/average first derivative of value e.g. strain rate - same dimensions as “value” (may be a vector)

**value_deriv2**: (optional) *NX_FLOAT* (Rank: same as field value, Dimensions: same as field value)
{units=*NX_ANY*}

Nominal/average second derivative of value - same dimensions as “value” (may be a vector)

**external_field_brief**: (optional) *NX_CHAR*

Any of these values:

- along beam
- across beam
- transverse
- solenoidal
- flow shear gradient
- flow vorticity

**depends_on**: (optional) *NX_CHAR*

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

**geometry**: (optional) *NXgeometry*

**DEPRECATED**: Use the field *depends_on* and *NXtransformations* to position the beamstop and NXoff_geometry to describe its shape instead

Defines the axes for logged vector quantities if they are not the global instrument axes.

**value_log**: (optional) *NXlog*

Time history of sensor readings

**value_deriv1_log**: (optional) *NXlog*

Time history of first derivative of sensor readings

**value_deriv2_log**: (optional) *NXlog*

Time history of second derivative of sensor readings

**external_field_full**: (optional) *NXorientation*

For complex external fields not satisfied by External_field_brief

**OFF_GEOMETRY**: (optional) *NXoff_geometry*

This group describes the shape of the sensor when necessary.

**TRANSFORMATIONS**: (optional) *NXtransformations*
This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXsensor/attached_to-field`
- `/NXsensor/depends_on-field`
- `/NXsensor/external_field_brief-field`
- `/NXsensor/external_field_full-group`
- `/NXsensor/geometry-group`
- `/NXsensor/high_trip_value-field`
- `/NXsensor/low_trip_value-field`
- `/NXsensor/measurement-field`
- `/NXsensor/model-field`
- `/NXsensor/name-field`
- `/NXsensor/OFF_GEOMETRY-group`
- `/NXsensor/run_control-field`
- `/NXsensor/short_name-field`
- `/NXsensor/TRANSFORMATIONS-group`
- `/NXsensor/type-field`
- `/NXsensor/value-field`
- `/NXsensor/value_deriv1-field`
- `/NXsensor/value_deriv1_log-group`
- `/NXsensor/value_deriv2-field`
- `/NXsensor/value_deriv2_log-group`
- `/NXsensor/value_log-group`
- `/NXsensor@default-attribute`

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/base_classes/NXsensor.nxdl.xml
**NXshape**

**Status:**

base class, extends *NXobject*

**Description:**

legacy class - (used by *NXgeometry*) - the shape and size of a component.

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

**Symbols:**

No symbol table

**Groups cited:**

none

**Structure:**

@default: (optional) *NX_CHAR*

Declares which child group contains a path leading to a *NXdata* group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See [https://www.nexusformat.org/2014_How_to_find_default_data.html](https://www.nexusformat.org/2014_How_to_find_default_data.html) for a summary of the discussion.

**shape:** (optional) *NX_CHAR*

general shape of a component

Any of these values:

- nxflat
- nxcylinder
- nxbox
- nxsphere
- nxcone
- nxelliptical
- nxtoroidal
- nxparabolic
- npolynomial

**size:** (optional) *NX_FLOAT* (Rank: 2, Dimensions: [numobj, nshapepar]) {units=*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 “nxcylinder” type the parameters are (diameter,height) and a three value orientation vector of the cylinder.
- For the “nxbox” type the parameters are (length,width,height).
- For the “nxsphere” type the parameters are (diameter).
- For nxcone cone half aperture

3.3. NeXus Class Definitions 301
• For nxelliptical, semi-major axis, semi-minor-axis, angle of major axis and pole
• For nxtoroidal, major radius, minor radius
• For nxparabolic, parabolic parameter a
• For nxpolygonal, an array of polynom coefficients, the dimension of the array encodes the degree of the polynom

direction: (optional) NX_CHAR
  Any of these values: concave | convex

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXshape/direction-field
- /NXshape/shape-field
- /NXshape/size-field
- /NXshape@default-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXshape.nxdl.xml

NXslit

Status:
  base class, extends NXobject

Description:
  A simple slit.
  For more complex geometries, NXaperture should be used.

Symbols:
  No symbol table

Groups cited:
  NXtransformations

Structure:
  @default: (optional) NX_CHAR
    Declares which child group contains a path leading to a NXdata group.
    It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.
  depends_on: (optional) NX_CHAR
    Points to the path of the last element in the geometry chain that places this object in space. When followed through that chain is supposed to end at an element depending on "." i.e. the origin of the coordinate system. If desired the location of the slit can also be described relative to an NXbeam, which will allow a simple description of a non-centred slit.
The reference plane of the slit is orthogonal to the z axis and includes the surface that is the entry surface of the slit. The reference point of the slit is the centre of the slit opening in the x and y axis on the reference plane. The reference point on the z axis is the reference plane.

\[ \text{x\_gap: (optional) } \text{NX\_NUMBER \{units=NX\_LENGTH\} } \]
Size of the gap opening in the first dimension of the local coordinate system.

\[ \text{y\_gap: (optional) } \text{NX\_NUMBER \{units=NX\_LENGTH\} } \]
Size of the gap opening in the second dimension of the local coordinate system.

**TRANSFORMATIONS:** (optional) **NXtransformations**

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXslit/depends_on-field
- /NXslit/TRANSFORMATIONS-group
- /NXslit/x\_gap-field
- /NXslit/y\_gap-field
- /NXslit/@default-attribute

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/base_classes/NXslit.nxdl.xml

**NXsource**

**Status:**
base class, extends **NXobject**

**Description:**
The neutron or x-ray storage ring/facility.

**Symbols:**
No symbol table

**Groups cited:**
**NXdata, NXgeometry, NXnote, NXoff Geometry, NXtransformations**

**Structure:**
@**default:** (optional) **NX\_CHAR**

3.3. NeXus Class Definitions
Declares which child group contains a path leading to a *NXdata* group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See [https://www.nexusformat.org/2014_How_to_find_default_data.html](https://www.nexusformat.org/2014_How_to_find_default_data.html) for a summary of the discussion.

**distance**: (optional) *NX_FLOAT* {units=*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**: (optional) *NX_CHAR*

Name of source

@**short_name**: (optional) *NX_CHAR*

short name for source, perhaps the acronym

**type**: (optional) *NX_CHAR*

type of radiation source (pick one from the enumerated list and spell exactly)

Any of these values:

- 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
- Ion Source
- UV Plasma Source
- Metal Jet X-ray

**probe**: (optional) *NX_CHAR*

type of radiation probe (pick one from the enumerated list and spell exactly)

Any of these values:

- neutron
- x-ray
- muon
- electron
- ultraviolet
- visible light
- positron
• proton

**power:** (optional) \textit{NX_FLOAT} \{units=\textit{NX\_POWER}\}

Source power

**emittance\_x:** (optional) \textit{NX_FLOAT} \{units=\textit{NX\_EMITTANCE}\}

Source emittance (nm-rad) in X (horizontal) direction.

**emittance\_y:** (optional) \textit{NX_FLOAT} \{units=\textit{NX\_EMITTANCE}\}

Source emittance (nm-rad) in Y (horizontal) direction.

**sigma\_x:** (optional) \textit{NX_FLOAT} \{units=\textit{NX\_LENGTH}\}

Particle beam size in x

**sigma\_y:** (optional) \textit{NX_FLOAT} \{units=\textit{NX\_LENGTH}\}

Particle beam size in y

**flux:** (optional) \textit{NX_FLOAT} \{units=\textit{NX\_FLUX}\}

Source intensity/area (example: s-1 cm-2)

**energy:** (optional) \textit{NX_FLOAT} \{units=\textit{NX\_ENERGY}\}

Source energy. For storage rings, this would be the particle beam energy. For X-ray tubes, this would be the excitation voltage.

**current:** (optional) \textit{NX_FLOAT} \{units=\textit{NX\_CURRENT}\}

Accelerator, X-ray tube, or storage ring current

**voltage:** (optional) \textit{NX_FLOAT} \{units=\textit{NX\_VOLTAGE}\}

Accelerator voltage

**frequency:** (optional) \textit{NX_FLOAT} \{units=\textit{NX\_FREQUENCY}\}

Frequency of pulsed source

**period:** (optional) \textit{NX_FLOAT} \{units=\textit{NX\_PERIOD}\}

Period of pulsed source

**target\_material:** (optional) \textit{NX\_CHAR}

Pulsed source target material

Any of these values:

• Ta

• W

• depleted\_U

• enriched\_U

• Hg

• Pb

• C

**number\_of\_bunches:** (optional) \textit{NX\_INT}

For storage rings, the number of bunches in use.

\textbf{3.3. NeXus Class Definitions}
bunch_length: (optional) \texttt{NX\_FLOAT \{units=NX\_TIME\}}

For storage rings, temporal length of the bunch

bunch_distance: (optional) \texttt{NX\_FLOAT \{units=NX\_TIME\}}

For storage rings, time between bunches

pulse_width: (optional) \texttt{NX\_FLOAT \{units=NX\_TIME\}}

temporal width of source pulse

mode: (optional) \texttt{NX\_CHAR}

source operating mode

Any of these values:

\begin{itemize}
  \item Single Bunch: for storage rings
  \item Multi Bunch: for storage rings
\end{itemize}

top_up: (optional) \texttt{NX\_BOOLEAN}

Is the synchrotron operating in top_up mode?

last_fill: (optional) \texttt{NX\_NUMBER \{units=NX\_CURRENT\}}

For storage rings, the current at the end of the most recent injection.

\@time: (optional) \texttt{NX\_DATE\_TIME}

date and time of the most recent injection.

depends_on: (optional) \texttt{NX\_CHAR}

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

The reference point of the source plane is its center in the x and y axis. The source is considered infinitely thin in the z axis.

notes: (optional) \texttt{NXnote}

any source/facility related messages/events that occurred during the experiment

bunch_pattern: (optional) \texttt{NXdata}

For storage rings, description of the bunch pattern. This is useful to describe irregular bunch patterns.

\textbf{title:} (optional) \texttt{NX\_CHAR}

name of the bunch pattern

pulse_shape: (optional) \texttt{NXdata}

source pulse shape
**geometry**: (optional) `NXgeometry`

**DEPRECATED**: Use the field `depends_on` and `NXtransformations` to position the source and `NXoff(geometry)` to describe its shape instead

“Engineering” location of source.

**OFF GEOMETRY**: (optional) `NXoff(geometry)`

This group describes the shape of the beam line component

**distribution**: (optional) `NXdata`

The wavelength or energy distribution of the source

**TRANSFORMATIONS**: (optional) `NXtransformations`

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

### Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXsource/bunch_distance-field`
- `/NXsource/bunch_length-field`
- `/NXsource/bunch_pattern-group`
- `/NXsource/bunch_pattern/title-field`
- `/NXsource/current-field`
- `/NXsource/depends_on-field`
- `/NXsource/distance-field`
- `/NXsource/distribution-group`
- `/NXsource/emittance_x-field`
- `/NXsource/emittance_y-field`
- `/NXsource/energy-field`
- `/NXsource/flux-field`
- `/NXsource/frequency-field`
- `/NXsource/geometry-group`
- `/NXsource/last_fill-field`
- `/NXsource/last_fill@time-attribute`
- `/NXsource/mode-field`
- `/NXsource/name-field`
- `/NXsource/name@short_name-attribute`
- `/NXsource/notes-group`
- `/NXsource/number_of_bunches-field`
- `/NXsource/OFF_GEOMETRY-group`
NXsubentry

Status:

base class, extends NXobject

Description:

Group of multiple application definitions for “multi-modal” (e.g. SAXS/WAXS) measurements.

NXsubentry is a base class virtually identical to NXentry and is used as the (overlay) location for application definitions. Use a separate NXsubentry for each application definition.

To use NXsubentry with a hypothetical application definition called NXmyappdef:

• Create a group with attribute NX_class="NXsubentry"
• Within that group, create a field called definition="NXmyappdef".
• There are two optional attributes of definition: version and URL

The intended use is to define application definitions for a multi-modal (a.k.a. multi-technique) NXentry. Previously, an application definition replaced NXentry with its own definition. With the increasing popularity of instruments combining multiple techniques for data collection (such as SAXS/WAXS instruments), it was recognized the application definitions must be entered in the NeXus data file tree as children of NXentry.

Symbols:

No symbol table

Groups cited:

NXcollection, NXdata, NXinstrument, NXmonitor, NXnote, NXparameters, NXprocess, NXsample, NXuser

Structure:

@default: (optional) NX_CHAR
Declares which \textit{NXdata} group contains the data to be shown by default. It is used to resolve ambiguity when one \textit{NXdata} group exists. The value \textit{names} the default \textit{NXentry} group. The value must be the name of a child of the current group. The child must be a NeXus group or a link to a NeXus group.

For more information about how NeXus identifies the default plottable data, see the \textit{Find Plottable Data, v3} section.

\texttt{@IDF\_Version}: (optional) \texttt{NX\_CHAR}

ISIS Muon IDF\_Version

title: (optional) \texttt{NX\_CHAR}

Extended title for entry

\texttt{experiment\_identifier}: (optional) \texttt{NX\_CHAR}

Unique identifier for the experiment, defined by the facility, possibly linked to the proposals

\texttt{experiment\_description}: (optional) \texttt{NX\_CHAR}

Brief summary of the experiment, including key objectives.

\texttt{collection\_identifier}: (optional) \texttt{NX\_CHAR}

User or Data Acquisition defined group of NeXus files or \textit{NXentry}

\texttt{collection\_description}: (optional) \texttt{NX\_CHAR}

Brief summary of the collection, including grouping criteria.

\texttt{entry\_identifier}: (optional) \texttt{NX\_CHAR}

Unique identifier for the measurement, defined by the facility.

\texttt{definition}: (optional) \texttt{NX\_CHAR}

Official NeXus NXDL schema to which this subentry conforms

\texttt{@version}: (optional) \texttt{NX\_CHAR}

NXDL version number

\texttt{@URL}: (optional) \texttt{NX\_CHAR}

URL of NXDL file

\texttt{definition\_local}: (optional) \texttt{NX\_CHAR}

Local NXDL schema extended from the subentry specified in the \texttt{definition} field. This contains any locally-defined, additional fields in the subentry.

\texttt{@version}: (optional) \texttt{NX\_CHAR}

NXDL version number

\texttt{@URL}: (optional) \texttt{NX\_CHAR}

URL of NXDL file

\texttt{start\_time}: (optional) \texttt{NX\_DATE\_TIME}

Starting time of measurement

\texttt{end\_time}: (optional) \texttt{NX\_DATE\_TIME}

Ending time of measurement

\texttt{duration}: (optional) \texttt{NX\_INT} \{units=\texttt{NX\_TIME}\}
Duration of measurement

**collection_time**: (optional) \texttt{NX\_FLOAT \{units=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**: (optional) \texttt{NX\_CHAR}

Such as “2007-3”. Some user facilities organize their beam time into run cycles.

**program_name**: (optional) \texttt{NX\_CHAR}

Name of program used to generate this file

**version**: (optional) \texttt{NX\_CHAR}

Program version number

**configuration**: (optional) \texttt{NX\_CHAR}

configuration of the program

**revision**: (optional) \texttt{NX\_CHAR}

Revision id of the file due to re-calibration, reprocessing, new analysis, new instrument definition format, …

**comment**: (optional) \texttt{NX\_CHAR}

**pre_sample_flightpath**: (optional) \texttt{NX\_FLOAT \{units=NX\_LENGTH\}}

This is the flightpath before the sample position. This can be determined by a chopper, by the moderator or the source itself. In other words: it the distance to the component which gives the T0 signal to the detector electronics. If another component in the NXinstrument hierarchy provides this information, this should be a link.

**experiment_documentation**: (optional) \texttt{NXnote}

Description of the full experiment (document in pdf, latex, …)

**notes**: (optional) \texttt{NXnote}

Notes describing entry

**thumbnail**: (optional) \texttt{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.

**mime_type**: (optional) \texttt{NX\_CHAR}

The value should be an \texttt{image/*}

Obligatory value: \texttt{image/*}

**USER**: (optional) \texttt{NXuser}

**SAMPLE**: (optional) \texttt{NXsample}

**INSTRUMENT**: (optional) \texttt{NXinstrument}

**COLLECTION**: (optional) \texttt{NXcollection}

**MONITOR**: (optional) \texttt{NXmonitor}

**DATA**: (optional) \texttt{NXdata}

**PARAMETERS**: (optional) \texttt{NXparameters}
**PROCESS**: (optional) *NXprocess*

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXsubentry/COLLECTION-group
- /NXsubentry/collection_description-field
- /NXsubentry/collection_identifier-field
- /NXsubentry/collection_time-field
- /NXsubentry/DATA-group
- /NXsubentry/definition-field
- /NXsubentry/definition@URL-attribute
- /NXsubentry/definition@version-attribute
- /NXsubentry/definition_local-field
- /NXsubentry/definition_local@URL-attribute
- /NXsubentry/definition_local@version-attribute
- /NXsubentry/duration-field
- /NXsubentry/end_time-field
- /NXsubentry/entry_identifier-field
- /NXsubentry/experiment_description-field
- /NXsubentry/experiment_documentation-group
- /NXsubentry/experiment_identifier-field
- /NXsubentry/INSTRUMENT-group
- /NXsubentry/MONITOR-group
- /NXsubentry/notes-group
- /NXsubentry/PARAMETERS-group
- /NXsubentry/pre_sample_flightpath-field
- /NXsubentry/PROCESS-group
- /NXsubentry/program_name-field
- /NXsubentry/program_name@configuration-attribute
- /NXsubentry/program_name@version-attribute
- /NXsubentry/revision-field
- /NXsubentry/revision@comment-attribute
- /NXsubentry/run_cycle-field
- /NXsubentry/SAMPLE-group
- /NXsubentry/start_time-field
- /NXsubentry/thumbnail-group
NXtransformations

Status:

base class, extends NXobject

Description:

Collection of axis-based translations and rotations to describe a geometry. May also contain axes that
do not move and therefore do not have a transformation type specified, but are useful in understanding
coordinate frames within which transformations are done, or in documenting important directions, such
as the direction of gravity.

A nested sequence of transformations lists the translation and rotation steps needed to describe the position
and orientation of any movable or fixed device.

There will be one or more transformations (axes) defined by one or more fields for each transformation.
Transformations can also be described by NXlog groups when the values change with time. The all-
caps name AXISNAME designates the particular axis generating a transformation (e.g. a rotation axis or a
translation axis or a general axis). The attribute units="NX_TRANSFORMATION" designates the units will
be appropriate to the transformation_type attribute:

- NX_LENGTH for translation
- NX_ANGLE for rotation
- NX_UNITLESS for axes for which no transformation type is specified

This class will usually contain all axes of a sample stage or goniometer or a detector. The NeXus default
McSTAS coordinate frame is assumed, but additional useful coordinate axes may be defined by using axes
for which no transformation type has been specified.

The entry point (depends_on) will be outside of this class and point to a field in here. Following the chain
may also require following depends_on links to transformations outside, for example to a common base
table. If a relative path is given, it is relative to the group enclosing the depends_on specification.

For a chain of three transformations, where $T_1$ depends on $T_2$ and that in turn depends on $T_3$, the final
transformation $T_f$ is

$$T_f = T_3 T_2 T_1$$

In explicit terms, the transformations are a subset of affine transformations expressed as 4x4 matrices that
act on homogeneous coordinates, $w = (x, y, z, 1)^T$.

For rotation and translation,

$$T_r = \begin{pmatrix} R & o \\ 0_3 & 1 \end{pmatrix}$$

$$T_t = \begin{pmatrix} I_3 & t + o \\ 0_3 & 1 \end{pmatrix}$$
where $R$ is the usual 3x3 rotation matrix, $o$ is an offset vector, $0_3$ is a row of 3 zeros, $I_3$ is the 3x3 identity matrix and $t$ is the translation vector.

$o$ is given by the offset attribute, $t$ is given by the vector attribute multiplied by the field value, and $R$ is defined as a rotation about an axis in the direction of vector, of angle of the field value.

**NOTE**

One possible use of NXtransformations is to define the motors and transformations for a diffractometer (goniometer). Such use is mentioned in the NXinstrument base class. Use one NXtransformations group for each diffractometer and name the group appropriate to the device. Collecting the motors of a sample table or xyz-stage in an NXtransformations group is equally possible.

Following the section on the general description of axis in NXtransformations is a section which documents the fields commonly used within NeXus for positioning purposes and their meaning. Whenever there is a need for positioning a beam line component please use the existing names. Use as many fields as needed in order to position the component. Feel free to add more axis if required. In the description given below, only those attributes which are defined through the name are specified. Add the other attributes of the full set:

- vector
- offset
- transformation_type
- depends_on

as needed.

**Symbols:**

No symbol table

**Groups cited:**

none

**Structure:**

@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

AXISNAME: (optional) NX_NUMBER {units=NX_TRANSFORMATION }

Units need to be appropriate for translation or rotation

The name of this field is not forced. The user is free to use any name that does not cause confusion. When using more than one AXISNAME field, make sure that each field name is unique in the same group, as required by HDF5.

The values given should be the start points of exposures for the corresponding frames. The end points should be given in AXISNAME_end.

@transformation_type: (optional) NX_CHAR

The transformation_type may be translation, in which case the values are linear displacements along the axis, rotation, in which case the values are angular rotations around the axis.
If this attribute is omitted, this is an axis for which there is no motion to be specifies, such as the direction of gravity, or the direction to the source, or a basis vector of a coordinate frame.

Any of these values: translation|rotation

@vector: (required) NX_NUMBER

Three values that define the axis for this transformation. The axis should be normalized to unit length, making it dimensionless. For rotation axes, the direction should be chosen for a right-handed rotation with increasing angle. For translation axes the direction should be chosen for increasing displacement. For general axes, an appropriate direction should be chosen.

@offset: (optional) NX_NUMBER

A fixed offset applied before the transformation (three vector components). This is not intended to be a substitute for a fixed translation axis but, for example, as the mechanical offset from mounting the axis to its dependency.

@offset_units: (optional) NX_CHAR

Units of the offset. Values should be consistent with NX_LENGTH.

@depends_on: (optional) NX_CHAR

Points to the path to a field defining the axis on which this depends or the string ‘.’.

@equipment_component: (optional) NX_CHAR

An arbitrary identifier of a component of the equipment to which the transformation belongs, such as ‘detector_arm’ or ‘detector_module’. NXtransformations with the same equipment_component label form a logical grouping which can be combined together into a single change-of-basis operation.

AXISNAME_end: (optional) NX_NUMBER {units=NX_TRANSFORMATION}

AXISNAME_end is a placeholder for a name constructed from the actual name of an axis to which _end has been appended.

The values in this field are the end points of the motions that start at the corresponding positions given in the AXISNAME field.

AXISNAME_increment_set: (optional) NX_NUMBER {units=NX_TRANSFORMATION}

AXISNAME_increment_set is a placeholder for a name constructed from the actual name of an axis to which _increment_set has been appended.

The value of this optional field is the intended average range through which the corresponding axis moves during the exposure of a frame. Ideally, the value of this field added to each value of AXISNAME would agree with the corresponding values of AXISNAME_end, but there is a possibility of significant differences. Use of AXISNAME_end is recommended.
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXtransformations/AXISNAME-field
- /NXtransformations/AXISNAME@depends_on-attribute
- /NXtransformations/AXISNAME@equipment_component-attribute
- /NXtransformations/AXISNAME@offset-attribute
- /NXtransformations/AXISNAME@offset_units-attribute
- /NXtransformations/AXISNAME@transformation_type-attribute
- /NXtransformations/AXISNAME@vector-attribute
- /NXtransformations/AXISNAME_end-field
- /NXtransformations/AXISNAME_increment_set-field
- /NXtransformations@default-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXtransformations.nxdl.xml

NXtranslation

Status:
base class, extends NXobject

Description:
legacy class - (used by NXgeometry) - general spatial location of a component.

Symbols:
No symbol table

Groups cited:
NXgeometry

Structure:
@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default
dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html
for a summary of the discussion.

distances: (optional) NX_FLOAT (Rank: 2, Dimensions: [numobj, 3]) {units=NX_LENGTH}
(x,y,z) This field describes the lateral movement of a component. The pair of groups
NXtranslation and NXorientation together 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.

gometry: (optional) NXgeometry

3.3. NeXus Class Definitions
Link to other object if we are relative, else absent

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXtranslation/distances-field`
- `/NXtranslation/geometry-group`
- `/NXtranslation@default-attribute`

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/base_classes/NXtranslation.nxdl.xml

**NXuser**

**Status:**

base class, extends *NXobject*

**Description:**

Contact information for a user.

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.

**Symbols:**

No symbol table

**Groups cited:**

none

**Structure:**

- **@default**: (optional) `NX_CHAR`
  
  Declares which child group contains a path leading to a *NXdata* group.

  It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

- **name**: (optional) `NX_CHAR`
  
  Name of user responsible for this entry

- **role**: (optional) `NX_CHAR`
  
  Role of user responsible for this entry. Suggested roles are “local_contact”, “principal_investigator”, and “proposer”

- **affiliation**: (optional) `NX_CHAR`
  
  Affiliation of user

- **address**: (optional) `NX_CHAR`
  
  Address of user

- **telephone_number**: (optional) `NX_CHAR`
  
  Telephone number of user
\textbf{fax\_number:} (optional) \textit{NX\_CHAR}

Fax number of user

\textbf{email:} (optional) \textit{NX\_CHAR}

Email of user

\textbf{facility\_user\_id:} (optional) \textit{NX\_CHAR}

facility based unique identifier for this person e.g. their identification code on the facility address/contact database

\textbf{ORCID:} (optional) \textit{NX\_CHAR}

an author code, Open Researcher and Contributor ID, defined by \url{https://orcid.org} and expressed as a URI

\noindent \textbf{Hypertext Anchors}

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXuser/address-field
- /NXuser/affiliation-field
- /NXuser/email-field
- /NXuser/facility_user_id-field
- /NXuser/fax_number-field
- /NXuser/name-field
- /NXuser/ORCID-field
- /NXuser/role-field
- /NXuser/telephone_number-field
- /NXuser@default-attribute

\noindent \textbf{NXDL Source:}

\url{https://github.com/nexusformat/definitions/blob/main/base_classes/NXuser.nxdl.xml}

\noindent \textbf{NXvelocity\_selector}

\noindent \textbf{Status:}

base class, extends \textit{NXobject}

\noindent \textbf{Description:}

A neutron velocity selector

\noindent \textbf{Symbols:}

No symbol table

\noindent \textbf{Groups cited:}

\textit{NXgeometry, NXoff\_geometry, NXtransformations}

\noindent \textbf{Structure:}

@\texttt{default}: (optional) \textit{NX\_CHAR}
Declares which child group contains a path leading to a *NXdata* group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See [https://www.nexusformat.org/2014_How_to_find_default_data.html](https://www.nexusformat.org/2014_How_to_find_default_data.html) for a summary of the discussion.

**type**: (optional) *NX_CHAR*

velocity selector type

**rotation_speed**: (optional) *NX_FLOAT* {units=*NX_FREQUENCY*}

velocity selector rotation speed

**radius**: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

radius at beam centre

**spwidth**: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

spoke width at beam centre

**length**: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

rotor length

**num**: (optional) *NX_INT* {units=*NX_UNITLESS*}

number of spokes/lamella

**twist**: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

twist angle along axis

**table**: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

offset vertical angle

**height**: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

input beam height

**width**: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

input beam width

**wavelength**: (optional) *NX_FLOAT* {units=*NX_WAVELENGTH*}

wavelength

**wavelength_spread**: (optional) *NX_FLOAT* {units=*NX_WAVELENGTH*}

deviation FWHM /Wavelength

**depends_on**: (optional) *NX_CHAR*

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

**geometry**: (optional) *NXgeometry*

DEPRECATED: Use the field *depends_on* and *NXtransformations* to position the velocity selector and NXoff_geometry to describe its shape instead

**OFF_GEOMETRY**: (optional) *NXoff_geometry*
This group describes the shape of the beam line component

**TRANSFORMATIONS:** (optional) *NXtransformations*

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

### Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXvelocity_selector/depends_on-field`
- `/NXvelocity_selector/geometry-group`
- `/NXvelocity_selector/height-field`
- `/NXvelocity_selector/length-field`
- `/NXvelocity_selector/num-field`
- `/NXvelocity_selector/OFF_GEOMETRY-group`
- `/NXvelocity_selector/radius-field`
- `/NXvelocity_selector/rotation_speed-field`
- `/NXvelocity_selector/spwidth-field`
- `/NXvelocity_selector/table-field`
- `/NXvelocity_selector/TRANSFORMATIONS-group`
- `/NXvelocity_selector/twist-field`
- `/NXvelocity_selector/type-field`
- `/NXvelocity_selector/wavelength-field`
- `/NXvelocity_selector/wavelength_spread-field`
- `/NXvelocity_selector/width-field`
- `/NXvelocity_selector@default-attribute`

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/base_classes/NXvelocity_selector.nxdl.xml

### NXxraylens

**Status:**

base class, extends *NXobject*

**Description:**

An X-ray lens, typically at a synchrotron X-ray beam line.

Based on information provided by Gerd Wellenreuther (DESY).

**Symbols:**

No symbol table
Groups cited:

NXnote, NXoff_geometry, NXtransformations

Structure:

@default: (optional) NX_CHAR

Declares which child group contains a path leading to a NXdata group.

It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

lens_geometry: (optional) NX_CHAR

Geometry of the lens

Any of these values:

• paraboloid
• spherical
• elliptical
• hyperbolical

symmetric: (optional) NX_BOOLEAN

Is the device symmetric?

cylindrical: (optional) NX_BOOLEAN

Is the device cylindrical?

focus_type: (optional) NX_CHAR

The type of focus of the lens

Any of these values: line | point

lens_thickness: (optional) NX_FLOAT {units=NX_LENGTH}

Thickness of the lens

lens_length: (optional) NX_FLOAT {units=NX_LENGTH}

Length of the lens

curvature: (optional) NX_FLOAT {units=NX_LENGTH}

Radius of the curvature as measured in the middle of the lens

aperture: (optional) NX_FLOAT {units=NX_LENGTH}

Diameter of the lens.

number_of_lenses: (optional) NX_INT

Number of lenses that make up the compound lens.

lens_material: (optional) NX_CHAR

Material used to make the lens.

gas: (optional) NX_CHAR

Gas used to fill the lens

gas_pressure: (optional) NX_FLOAT {units=NX_PRESSURE}
Gas pressure in the lens

**depends_on**: (optional) *NX_CHAR*

NeXus positions components by applying a set of translations and rotations to apply to the component starting from 0, 0, 0. The order of these operations is critical and forms what NeXus calls a dependency chain. The depends_on field defines the path to the top most operation of the dependency chain or the string “.” if located in the origin. Usually these operations are stored in a NXtransformations group. But NeXus allows them to be stored anywhere.

**cylinder_orientation**: (optional) *NXnote*

Orientation of the cylinder axis.

**OFF_GEOMETRY**: (optional) *NXoff_geometry*

This group describes the shape of the beam line component

**TRANSFORMATIONS**: (optional) *NXtransformations*

This is the group recommended for holding the chain of translation and rotation operations necessary to position the component within the instrument. The dependency chain may however traverse similar groups in other component groups.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXxraylens/aperture-field
- /NXxraylens/curvature-field
- /NXxraylens/cylinder_orientation-group
- /NXxraylens/cylindrical-field
- /NXxraylens/depends_on-field
- /NXxraylens/focus_type-field
- /NXxraylens/gas-field
- /NXxraylens/gas_pressure-field
- /NXxraylens/lens_geometry-field
- /NXxraylens/lens_length-field
- /NXxraylens/lens_material-field
- /NXxraylens/lens_thickness-field
- /NXxraylens/number_of_lenses-field
- /NXxraylens/OFF_GEOMETRY-group
- /NXxraylens/symmetric-field
- /NXxraylens/TRANSFORMATIONS-group
- /NXxraylens/default-attribute

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/base_classes/NXxraylens.nxdl.xml
3.3.2 Application Definitions

A description of each NeXus application definition is given. NeXus application definitions define the minimum set of terms that must be used in an instance of that class. Application definitions also may define terms that are optional in the NeXus data file. The definition, in this case, reserves the exact term by declaring its spelling and description. Consider an application definition as a contract between a data provider (such as the beam line control system) and a data consumer (such as a data analysis program for a scientific technique) that describes the information is certain to be available in a data file.

Use NeXus links liberally in data files to reduce duplication of data. In application definitions involving raw data, write the raw data in the NXinstrument tree and then link to it from the location(s) defined in the relevant application definition.

NXarchive
This is a definition for data to be archived by ICAT (http://www.icatproject.org/).

NXarps
This is an application definition for angular resolved photo electron spectroscopy.

NXcanSAS
Implementation of the canSAS standard to store reduced small-angle scattering data of any dimension.

NXdirecttof
This is a application definition for raw data from a direct geometry TOF spectrometer

NXfluo
This is an application definition for raw data from an X-ray fluorescence experiment

NXindirecttof
This is an application definition for raw data from a direct geometry TOF spectrometer

NXiqproc
Application definition for any \( I(Q) \) data.

NXlauetof
This is the application definition for a TOF laue diffractometer

NXmonopd
Monochromatic Neutron and X-Ray Powder diffractometer

NXmx
functional application definition for macromolecular crystallography

NXrefscan
This is an application definition for a monochromatic scanning reflectometer.

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

NXsas
Raw, monochromatic 2-D SAS data with an area detector.

NXsastof
raw, 2-D SAS data with an area detector with a time-of-flight source

NXscan
Application definition for a generic scan instrument.

NXspe
NXSPE Inelastic Format. Application definition for NXSPE file format.

NXsqom
This is the application definition for S(Q,OM) processed data.
NXstxm  
Application definition for a STXM instrument.

NXtas  
This is an application definition for a triple axis spectrometer.

NXtofnpd  
This is an application definition for raw data from a TOF neutron powder diffractometer

NXtofraw  
This is an application definition for raw data from a generic TOF instrument

NXtofsingle  
This is a application definition for raw data from a generic TOF instrument

NXtomo  
This is the application definition for x-ray or neutron tomography raw data.

NXtomophase  
This is the application definition for x-ray or neutron tomography raw data with phase contrast variation at each point.

NXtomoproc  
This is an application definition for the final result of a tomography experiment: a 3D construction of some volume of physical properties.

NXxas  
This is an application definition for raw data from an X-ray absorption spectroscopy experiment.

NXxasproc  
Processed data from XAS. This is energy versus I(incoming)/I(absorbed).

NXxbase  
This definition covers the common parts of all monochromatic single crystal raw data application definitions.

NXxeuler  
raw data from a four-circle diffractometer with an eulerian cradle, extends NXxbase

NXxkappa  
raw data from a kappa geometry (CAD4) single crystal diffractometer, extends NXxbase

NXxlaue  
raw data from a single crystal laue camera, extends NXxrot

NXxlaueplate  
raw data from a single crystal Laue camera, extends NXxlaue

NXxnb  
raw data from a single crystal diffractometer, extends NXxbase

NXxrot  
raw data from a rotation camera, extends NXxbase
NXarchive

Status:
application definition, extends NXobject

Description:
This is a definition for data to be archived by ICAT (http://www.icatproject.org/).

Symbols:
No symbol table

Groups cited:
NXentry, NXinstrument, NXsample, NXsource, NXuser

Structure:

**entry**: (required) NXentry

  @index: (required) NX_CHAR

  title: (required) NX_CHAR

  **experiment_identifier**: (required) NX_CHAR

    unique identifier for the experiment

  **experiment_description**: (required) NX_CHAR

    Brief description of the experiment and its objectives

  **collection_identifier**: (required) NX_CHAR

    ID of user or DAQ define group of data files

  **collection_description**: (required) NX_CHAR

    Brief summary of the collection, including grouping criteria

  **entry_identifier**: (required) NX_CHAR

    unique identifier for this measurement as provided by the facility

  **start_time**: (required) NX_DATE_TIME

  **end_time**: (required) NX_DATE_TIME

  **duration**: (required) NX_FLOAT {units=NX_TIME}

    TODO: needs documentation

  **collection_time**: (required) NX_FLOAT {units=NX_TIME}

    TODO: needs documentation

  **run_cycle**: (required) NX_CHAR

    TODO: needs documentation

  **revision**: (required) NX_CHAR

    revision ID of this file, may be after recalibration, reprocessing etc.

  **definition**: (required) NX_CHAR

    Official NeXus NXDL schema to which this file conforms

    Obligatory value: NXarchive
**program**: (required) *NX_CHAR*

The program and version used for generating this file

@**version**: (required) *NX_CHAR*

**release_date**: (required) *NX_CHAR* \{units=*NX_TIME*\}

when this file is to be released into PD

**user**: (required) *NXuser*

  **name**: (required) *NX_CHAR*

  **role**: (required) *NX_CHAR*

  role of the user

  **facility_user_id**: (required) *NX_CHAR*

  ID of the user in the facility bureaucracy database

**instrument**: (required) *NXinstrument*

  **name**: (required) *NX_CHAR*

  **description**: (required) *NX_CHAR*

  Brief description of the instrument

**SOURCE**: (required) *NXsource*

  **type**: (required) *NX_CHAR*

  Any of these values:

  • 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

  **name**: (required) *NX_CHAR*

  **probe**: (required) *NX_CHAR*

  Any of these values: neutron | x-ray | electron

**sample**: (required) *NXsample*

  **name**: (required) *NX_CHAR*

  Descriptive name of sample

  **sample_id**: (required) *NX_CHAR*

  Unique database id of the sample

  **description**: (required) *NX_CHAR*

  **type**: (required) *NX_CHAR*

  Any of these values:
• sample
• sample+can
• calibration sample
• normalisation sample
• simulated data
• none
• sample_environment

**chemical_formula: (required) NX_CHAR**

Chemical formula formatted according to CIF conventions

**preparation_date: (required) NX_CHAR {units=NX_TIME}**

**situation: (required) NX_CHAR**

Description of the environment the sample is in: air, vacuum, oxidizing atmosphere, dehydrated, etc.

**temperature: (required) NX_FLOAT {units=NX_TEMPERATURE}**

**magnetic_field: (required) NX_FLOAT {units=NX_CURRENT}**

**electric_field: (required) NX_FLOAT {units=NX_VOLTAGE}**

**stress_field: (required) NX_FLOAT {units=NX_UNITLESS}**

**pressure: (required) NX_FLOAT {units=NX_PRESSURE}**

---

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

• /NXarchive/entry-group
• /NXarchive/entry/collection_description-field
• /NXarchive/entry/collection_identifier-field
• /NXarchive/entry/collection_time-field
• /NXarchive/entry/definition-field
• /NXarchive/entry/duration-field
• /NXarchive/entry/end_time-field
• /NXarchive/entry/entry_identifier-field
• /NXarchive/entry/experiment_description-field
• /NXarchive/entry/experiment_identifier-field
• /NXarchive/entry/instrument-group
• /NXarchive/entry/instrument/description-field
• /NXarchive/entry/instrument/name-field
• /NXarchive/entry/instrument/SOURCE-group
• /NXarchive/entry/instrument/SOURCE/name-field
• /NXarchive/entry/instrument/SOURCE/probe-field
• /NXarchive/entry/instrument/SOURCE/type-field
• /NXarchive/entry/program-field
• /NXarchive/entry/program@version-attribute
• /NXarchive/entry/release_date-field
• /NXarchive/entry/revision-field
• /NXarchive/entry/run_cycle-field
• /NXarchive/entry/sample-group
• /NXarchive/entry/sample/chemical_formula-field
• /NXarchive/entry/sample/description-field
• /NXarchive/entry/sample/electric_field-field
• /NXarchive/entry/sample/magnetic_field-field
• /NXarchive/entry/sample/name-field
• /NXarchive/entry/sample/preparation_date-field
• /NXarchive/entry/sample/pressure-field
• /NXarchive/entry/sample/sample_id-field
• /NXarchive/entry/sample/situation-field
• /NXarchive/entry/sample/stress_field-field
• /NXarchive/entry/sample/temperature-field
• /NXarchive/entry/sample/type-field
• /NXarchive/entry/start_time-field
• /NXarchive/entry/title-field
• /NXarchive/entry/user-group
• /NXarchive/entry/user/facility_user_id-field
• /NXarchive/entry/user/name-field
• /NXarchive/entry/user/role-field
• /NXarchive/entry@index-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXarchive.nxdl.xml
NXarpes

Status:
application definition, extends NXobject

Description:
This is an application definition for angular resolved photo electron spectroscopy.
It has been drawn up with hemispherical electron analysers in mind.

Symbols:
No symbol table

Groups cited:
NXdata, NXdetector, NXentry, NXinstrument, NXmonochromator, NXsample, NXsource

Structure:
ENTRY: (required) NXentry

@entry: (required) NX_CHAR

NeXus convention is to use “entry1”, “entry2”,… for analysis software to locate each entry.

title: (required) NX_CHAR

start_time: (required) NX_DATE_TIME

definition: (required) NX_CHAR

Official NeXus NXDL schema to which this file conforms.

Obligatory value: NXarpes

INSTRUMENT: (required) NXinstrument

SOURCE: (required) NXsource

type: (required) NX_CHAR

name: (required) NX_CHAR

probe: (required) NX_CHAR

Obligatory value: x-ray

monochromator: (required) NXmonochromator

energy: (required) NX_NUMBER [units=NX_ENERGY]

analyser: (required) NXdetector

data: (required) NX_NUMBER

lens_mode: (required) NX_CHAR

setting for the electron analyser lens

acquisition_mode: (required) NX_CHAR

Any of these values: swept | fixed

entrance_slit_shape: (required) NX_CHAR

Any of these values: curved | straight
entrance_slit_setting: (required) NX_NUMBER {units=NX_ANY}
   dial setting of the entrance slit

entrance_slit_size: (required) NX_NUMBER {units=NX_LENGTH}
   size of the entrance slit

pass_energy: (required) NX_NUMBER {units=NX_ENERGY}
   energy of the electrons on the mean path of the analyser

time_per_channel: (required) NX_NUMBER {units=NX_TIME}
   todo: define more clearly

angles: (required) NX_NUMBER {units=NX_ANGLE}
   Angular axis of the analyser data which dimension the axis applies to is
defined using the normal NXdata methods.

energies: (required) NX_NUMBER {units=NX_ENERGY}
   Energy axis of the analyser data which dimension the axis applies to is
defined using the normal NXdata methods.

sensor_size: (required) NX_INT (Rank: 1, Dimensions: [2])
   number of raw active elements in each dimension

region_origin: (required) NX_INT (Rank: 1, Dimensions: [2])
   origin of rectangular region selected for readout

region_size: (required) NX_INT (Rank: 1, Dimensions: [2])
   size of rectangular region selected for readout

SAMPLE: (required) NXsample
   name: (required) NX_CHAR
      Descriptive name of sample

   temperature: (required) NX_NUMBER {units=NX_TEMPERATURE}

DATA: (required) NXdata

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

• /NXarpes/ENTRY-group
• /NXarpes/ENTRY/DATA-group
• /NXarpes/ENTRY/definition-field
• /NXarpes/ENTRY/INSTRUMENT-group
• /NXarpes/ENTRY/INSTRUMENT/analyser-group
• /NXarpes/ENTRY/INSTRUMENT/analyser/acquisition_mode-field
• /NXarpes/ENTRY/INSTRUMENT/analyser/angles-field
• /NXarpes/ENTRY/INSTRUMENT/analyser/data-field
NXcanSAS

Status:
application definition, extends NXobject

Description:
Implementation of the canSAS standard to store reduced small-angle scattering data of any dimension.

For more details, see:
- http://www.cansas.org/
- http://www.cansas.org/formats/canSAS1d/1.1/doc/
- https://github.com/canSAS-org/NXcanSAS_examples

The minimum requirements for reduced small-angle scattering data as described by canSAS are summarized in the following figure:
Implementation of canSAS standard in NeXus

This application definition is an implementation of the canSAS standard for storing both one-dimensional and multi-dimensional reduced small-angle scattering data.

- NXcanSAS is for reduced SAS data and metadata to be stored together in one file.
- *Reduced* SAS data consists of $I(Q)$ or $I(|Q|)$
- External file links are not to be used for the reduced data.
- A good practice/practise is, at least, to include a reference to how the data was acquired and processed. Yet this is not a requirement.
- There is no need for NXcanSAS to refer to any raw data.

The canSAS data format has a structure similar to NeXus, not identical. To allow canSAS data to be expressed in NeXus, yet identifiable by the canSAS standard, an additional group attribute `canSAS_class` was introduced. Here is the mapping of some common groups.

<table>
<thead>
<tr>
<th>group (*)</th>
<th>NX_class</th>
<th>canSAS_class</th>
</tr>
</thead>
<tbody>
<tr>
<td>sasentry</td>
<td>NXentry</td>
<td>SASentry</td>
</tr>
<tr>
<td>sasdata</td>
<td>NXdata</td>
<td>SASdata</td>
</tr>
<tr>
<td>sasdetector</td>
<td>NXdetector</td>
<td>SASdetector</td>
</tr>
<tr>
<td>sasinstrument</td>
<td>NXinstrument</td>
<td>SASinstrument</td>
</tr>
<tr>
<td>sasnote</td>
<td>NXnote</td>
<td>SASnote</td>
</tr>
<tr>
<td>sasprocess</td>
<td>NXprocess</td>
<td>SASprocess</td>
</tr>
<tr>
<td>sasprocessnote</td>
<td>NXcollection</td>
<td>SASprocessnote</td>
</tr>
<tr>
<td>sstransmission</td>
<td>NXdata</td>
<td>SASTransmission_spectrum</td>
</tr>
<tr>
<td>sassample</td>
<td>NXsample</td>
<td>SASsample</td>
</tr>
<tr>
<td>sassource</td>
<td>NXsource</td>
<td>SASsource</td>
</tr>
</tbody>
</table>

(*) The name of each group is a suggestion, not a fixed requirement and is chosen as fits each data file. See the section on defining *NXDL group and field names*.

Refer to the NeXus Coordinate System drawing (*The NeXus Coordinate System*) for choice and direction of $x$, $y$, and $z$ axes.
The minimum required information for a NeXus data file written to the NXcanSAS specification.

```plaintext
NXcanSAS HDF5 data file
entry : NXentry
    @NX_class = "NXentry"
    @canSAS_class = "SASentry"
    @version = "1.0"
    definition = "NXcanSAS"
    run = "<see the documentation>"
    title = "something descriptive yet short"
    data : NXdata
        @NX_class = "NXdata"
        @canSAS_class = "SASdata"
        @signal = "I"
        @I_axes = "<see the documentation>"
        @Q_indices : NX_INT = <see the documentation>
        I : NX_NUMBER
            @units = <see the documentation>
        Q : NX_NUMBER
            @units = NX_PER_LENGTH
```

Symbols:

No symbol table

Groups cited:

NXaperture, NXcollection, NXcollimator, NXdata, NXdetector, NXentry, NXinstrument, NXnote, NXprocess, NXsample, NXsource

Structure:

**ENTRY**: (required) **NXentry**

Place the canSAS SASentry group as a child of a NeXus NXentry group (when data from multiple techniques are being stored) or as a replacement for the NXentry group.

Note: It is required for all numerical objects to provide a **units** attribute that describes the engineering units. Use the Unidata UDunits specification as this is compatible with various community standards.

@**default**: (optional) **NX_CHAR**

Declares which **NXdata** group contains the data to be shown by default. It is needed to resolve ambiguity when more than one **NXdata** group exists. The value is the name of the default **NXdata** group. Usually, this will be the name of the first SASdata group.

@**canSAS_class**: (required) **NX_CHAR**

Official canSAS group: SASentry

Obligatory value: SASentry

@**version**: (required) **NX_CHAR**

Describes the version of the canSAS standard used to write this data. This must be a text (not numerical) representation. Such as:

---

1 The UDunits specification also includes instructions for derived units.
@version="1.1"

Obligatory value: 1.1

definition: (required) NX_CHAR

Official NeXus NXDL schema to which this subentry conforms.

Obligatory value: NXcanSAS

title: (required) NX_CHAR

Title of this SASentry. Make it so that you can recognize the data by its title. Could be the name of the sample, the name for the measured data, or something else representative.

run: (required) NX_CHAR

Run identification for this SASentry. For many facilities, this is an integer, such as an experiment number. Use multiple instances of run as needed, keeping in mind that HDF5 requires unique names for all entities in a group.

@name: (optional) NX_CHAR

Optional string attribute to identify this particular run. Could use this to associate (correlate) multiple SASdata elements with run elements.

DATA: (required) NXdata

A SASData group contains a single reduced small-angle scattering data set that can be represented as $I(\vec{Q})$ or $I(|\vec{Q}|)$.

$Q$ can be either a vector ($\vec{Q}$) or a vector magnitude ($|\vec{Q}|$).

The name of each SASdata group must be unique within a SASentry group. Suggest using names such as sasdata01.

NOTE: For the first SASdata group, be sure to write the chosen name into the SASentry/@default attribute, as in:

SASentry/@default="sasdata01"

A SASdata group has several attributes:

- I_axes
- Q_indices
- Mask_indices

To indicate the dependency relationships of other varied parameters, use attributes similar to @Mask_indices (such as @Temperature_indices or @Pressure_indices).

@canSAS_class: (required) NX_CHAR

Official canSAS group: NXcanSAS (applications); SASdata

Obligatory value: SASdata

@signal: (required) NX_CHAR

Name of the default data field.

Obligatory value:
• I: For canSAS SASdata, this is always “I”.

@I_axes: (required) NX_CHAR

String array that defines the independent data fields used in the default plot for all of the dimensions of the signal field (the signal field is the field in this group that is named by the signal attribute of this group). One entry is provided for every dimension of the I data object. Such as:

```
@I_axes="Temperature", "Time", "Pressure", "Q", "Q"
```

Since there are five items in the list, the intensity field of this example I must be a five-dimensional array (rank=5).

@Q_indices: (required) NX_INT

Integer or integer array that describes which indices (of the I data object) are used to reference the Q data object. The items in this array use zero-based indexing. Such as:

```
@Q_indices=1,3,4
```

which indicates that Q requires three indices from the I data object: one for time and two for Q position. Thus, in this example, the Q data is time-dependent: \( \vec{Q}(t) \).

@mask: (required) NX_CHAR

Name of the data mask field.

The data mask must have the same shape as the data field. Positions in the mask correspond to positions in the data field. The value of the mask field may be either a boolean array where false means no mask and true means mask or a more descriptive array as as defined in NXdetector.

@Mask_indices: (optional) NX_CHAR

Integer or integer array that describes which indices (of the I data object) are used to reference the Mask data object. The items in this array use zero-based indexing. Such as:

```
@Mask_indices=3,4
```

which indicates that Q requires two indices from the I data object for Q position.

@timestamp: (optional) NX_DATE_TIME

ISO-8601 time\(^2\)

Q: (required) NX_NUMBER \{ units=NX_PER_LENGTH \}

Array of Q data to accompany I.

Q may be represented as either the three-dimensional scattering vector \( \vec{Q} \) or the magnitude of the scattering vector, \( |\vec{Q}| \):

\[
|\vec{Q}| = \frac{4\pi}{\lambda} \sin(\theta)
\]

---

\(^2\) ISO-8601 standard time representation. NeXus dates and times are reported in ISO-8601 (e.g., yyyy-mm-ddTh:mm:ss) or modified ISO-8601 (e.g., yyyy-mm-dd hh:mm:ss). See: http://www.w3.org/TR/NOTE-datetime or http://en.wikipedia.org/wiki/ISO_8601 for more details.
When we write $Q$, we may refer to either or both of $|\vec{Q}|$ or $\vec{Q}$, depending on the context.

@units: (required) *NX_CHAR*

Engineering units to use when expressing $Q$ and related terms.

Data expressed in other units will generate a warning from validation software and may not be processed by some analysis software packages.

Any of these values:
- $1/m$
- $1/nm$: preferred
- $1/\text{angstrom}$

@uncertainties: (optional) *NX_CHAR*

(optional: for numerical arrays)

Names the dataset (in this SAS data group) that provides the uncertainty to be used for data analysis. The name of the dataset containing the $Q$ uncertainty is flexible. The name must be unique in the SAS data group.

Such as:

```
@uncertainties="Q_uncertainties"
```

The *uncertainties* field will have the same shape (dimensions) as the Q field.

These values are the estimates of uncertainty of each $Q$. By default, this will be interpreted to be the estimated standard deviation. In special cases, when a standard deviation cannot possibly be used, its value can specify another measure of distribution width.

There may also be a subdirectory (optional) with constituent components.

**Note:** To report distribution in reported $Q$ values, use the
@resolutions attribute. It is possible for both @resolutions and uncertainties to be reported.

@resolutions: (optional) NX_CHAR

(optional: for numerical arrays)

Names the dataset (in this SASdata group) containing the Q resolution. The name of the dataset containing the Q resolution is flexible. The name must be unique in the SASdata group.

The resolutions field will have the same shape (dimensions) as the Q field.

Generally, this is the principal resolution of each Q. Names the data object (in this SASdata group) that provides the Q resolution to be used for data analysis. Such as:

```
@resolutions="Qdev"
```

To specify two-dimensional resolution for slit-smearing geometry, such as \((dQ_w, dQ_l)\), use a string array, such as:

```
@resolutions="dQ_w", "dQ_l"
```

There may also be a subdirectory (optional) with constituent components.

This pattern will demonstrate how to introduce further as-yet unanticipated terms related to the data.

By default, the values of the resolutions data object are assumed to be one standard deviation of any function used to approximate the resolution function. This equates to the width of the gaussian distribution if a Gaussian is chosen. See the @resolutions_description attribute.

**Note:** To report uncertainty in reported Q values, use the @uncertainties attribute. It is possible for both @resolutions and uncertainties to be reported.

@resolutions_description: (optional) NX_CHAR

(optional) Generally, this describes the Q @resolutions data object. By default, the value is assumed to be “Gaussian”. These are suggestions:

- Gaussian
- Lorentzian
- Square: note that the full width of the square would be \(\sim 2.9\) times the standard deviation specified in the vector
- Triangular
- Sawtooth-outward: vertical edge pointing to larger Q
- Sawtooth-inward vertical edge pointing to smaller Q
• Bin: range of values contributing (for example, when 2-D detector data have been reduced to a 1-D \( I(|Q|) \) dataset)

For other meanings, it may be necessary to provide further details such as the function used to assess the resolution. In such cases, use additional datasets or a NXnote subgroup to include that detail.

I: (required) **NX_NUMBER**

Array of intensity \((I)\) data.

The intensity may be represented in one of these forms:

**absolute units**: \(d\Sigma/d\Omega(Q)\) differential cross-section per unit volume per unit solid angle (such as: \(1/cm/sr\) or \(1/m/sr\))

**absolute units**: \(d\sigma/d\Omega(Q)\) differential cross-section per unit atom per unit solid angle (such as: \(cm^2\) or \(m^2\))

**arbitrary units**: \(I(Q)\) usually a ratio of two detectors but units are meaningless (such as: a.u. or counts)

This presents a few problems for analysis software to sort out when reading the data. Fortunately, it is possible to analyze the units to determine which type of intensity is being reported and make choices at the time the file is read. But this is an area for consideration and possible improvement.

One problem arises with software that automatically converts data into some canonical units used by that software. The software should not convert units between these different types of intensity indiscriminately.

A second problem is that when arbitrary units are used, then the set of possible analytical results is restricted. With such units, no meaningful volume fraction or number density can be determined directly from \(I(Q)\).

In some cases, it is possible to apply a factor to convert the arbitrary units to an absolute scale. This should be considered as a possibility of the analysis process.

Where this documentation says *typical units*, it is possible that small-angle data may be presented in other units and still be consistent with NeXus. See the NeXus Data Units section.

@units: (required) **NX_CHAR**

Engineering units to use when expressing \(I\) and intensity-related terms.

Data expressed in other units (or missing a @units attribute) will be treated as arbitrary by some software packages.

For software using the UDUNITS-2 library, arbitrary will be changed to unknown for handling with that library.

Any of these values:

• 1/m: includes m2/m3 and 1/m/sr
• 1/cm: includes cm2/cm3 and 1/cm/sr
• m2/g
• cm2/g
• arbitrary
@uncertainties: (optional) NX_CHAR

(optional: for numerical arrays)

Names the dataset (in this SAS data group) that provides the uncertainty of I to be used for data analysis. The name of the dataset containing the I uncertainty is flexible. The name must be unique in the SAS data group.

Generally, this is the estimate of the uncertainty of each I. Typically the estimated standard deviation.

Idev is the canonical name from the 1D standard. The NXcanSAS standard allows for the name to be described using this attribute. Such as:

```
@uncertainties="Idev"
```

@scaling_factor: (optional) NX_CHAR

(optional) Names the field (a.k.a. dataset) that contains a factor to multiply I. By default, this value is unity. Should an uncertainty be associated with the scaling factor field, the field containing that uncertainty would be designated via the uncertainties attribute. Such as:

```
I : NX_NUMBER
   @uncertainties="Idev" : NX_CHAR
   @scaling_factor="I_scaling" : NX_CHAR
Idev : NX_NUMBER
I_scaling : NX_NUMBER
   @uncertainties="I_scaling_dev" : NX_CHAR
I_scaling_dev : NX_NUMBER
```

The exact names for I_scaling and I_scaling_dev are not defined by NXcanSAS. The user has the flexibility to use names different than those shown in this example.

Idev: (optional) NX_NUMBER

Estimated uncertainty (usually standard deviation) in I. Must have the same units as I.

When present, the name of this field is also recorded in the uncertainties attribute of I, as in:

```
I/@uncertainties="Idev"
```

@units: (required) NX_CHAR

Engineering units to use when expressing I and intensity-related terms.

Data expressed in other units (or missing a @units attribute) will generate a warning from any validation process and will be treated as arbitrary by some analysis software packages.

For software using the UDUNITS-2 library, arbitrary will be changed to unknown for handling with that library.

Any of these values:

- 1/m: includes m²/m³ and 1/m/sr
- 1/cm: includes cm²/cm³ and 1/cm/sr
• m\(^2\)/g
• cm\(^2\)/g
• arbitrary

\textbf{Qdev}: (optional) \texttt{NX_NUMBER \{units=NX\_PER\_LENGTH\}}

Estimated \(Q\) \textbf{resolution} (usually standard deviation). Must have the same units as \(Q\).

When present, the name of this field is also recorded in the \texttt{resolutions} attribute of \(Q\), as in:

\begin{verbatim}
Q/@resolutions="Qdev"
\end{verbatim}

or:

\begin{verbatim}
Q/@resolutions="dQw", "dQl"
\end{verbatim}

\texttt{@units}: (required) \texttt{NX\_CHAR}

Engineering units to use when expressing \(Q\) and related terms.

Data expressed in other units may not be processed by some software packages.

Any of these values:

• 1/m
• 1/nm: preferred
• 1/angstrom

\textbf{dQw}: (optional) \texttt{NX\_NUMBER \{units=NX\_PER\_LENGTH\}}

\(Q\) \textbf{resolution} along the axis of scanning (the high-resolution \textit{slit width} direction). Useful for defining resolution data from slit-smearing instruments such as Bonse-Hart geometry. Must have the same units as \(Q\).

When present, the name of this field is also recorded in the \texttt{resolutions} attribute of \(Q\), as in:

\begin{verbatim}
Q/@resolutions="dQw", "dQl"
\end{verbatim}

\texttt{@units}: (required) \texttt{NX\_CHAR}

Engineering units to use when expressing \(Q\) and related terms.

Data expressed in other units may not be processed by some software packages.

Any of these values:

• 1/m
• 1/nm: preferred
• 1/angstrom

\textbf{dQl}: (optional) \texttt{NX\_NUMBER \{units=NX\_PER\_LENGTH\}}
\( Q \) resolution perpendicular to the axis of scanning (the low-resolution slit length direction). Useful for defining resolution data from slit-smearing instruments such as Bonse-Hart geometry. Must have the same units as \( Q \).

When present, the name of this field is also recorded in the resolutions attribute of \( Q \), as in:

\[ Q/@resolutions=“dQw”, “dQl” \]

@units: (required) \textit{NX_CHAR}

Engineering units to use when expressing \( Q \) and related terms.

Data expressed in other units may not be processed by some software packages.

Any of these values:

- \( 1/m \)
- \( 1/nm \): preferred
- \( 1/\text{angstrom} \)

\( Q_{\text{mean}} \): (optional) \textit{NX_NUMBER} \{units=\textit{NX_PER_LENGTH}\}

Mean value of \( Q \) for this data point. Useful when describing data that has been binned from higher-resolution data.

It is expected that \( Q \) is provided and that both \( Q \) and \( Q_{\text{mean}} \) will have the same units.

@units: (required) \textit{NX_CHAR}

Engineering units to use when expressing \( Q \) and related terms.

Data expressed in other units may not be processed by some software packages.

Any of these values:

- \( 1/m \)
- \( 1/nm \): preferred
- \( 1/\text{angstrom} \)

\( \text{ShadowFactor} \): (optional) \textit{NX_CHAR} \{units=\textit{NX_DIMENSIONLESS}\}

A numerical factor applied to pixels affected by the beam stop penumbra. Used in data files from NIST/NCNR instruments.


\textbf{INSTRUMENT}: (optional) \textit{NXinstrument}

Description of the small-angle scattering instrument.

Consider, carefully, the relevance to the SAS data analysis process when adding subgroups in this \textit{NXinstrument} group. Additional information can be added but will likely be ignored by standardized data analysis processes.

The NeXus \textit{NXbeam} base class may be added as a subgroup of this \textit{NXinstrument} group or as a subgroup of the \textit{NXsample} group to describe properties of the beam at any point downstream from the source.

@canSAS_class: (required) \textit{NX_CHAR}
Official canSAS group: NXcanSAS (applications); SASinstrument

Obligatory value: SASinstrument

**APERTURE**: (optional) **NXaperture**

*NXaperture* is generic and limits the variation in data files.
Possible NeXus base class alternatives are: *NXpinhole* or *NXslit*.

@canSAS_class: (required) **NX_CHAR**

Official canSAS group: NXcanSAS (applications); SASaperture

Obligatory value: SASaperture

**shape**: (required) **NX_CHAR**

describe the type of aperture (pinhole, 4-blade slit, Soller slit, …)

**x_gap**: (optional) **NX_NUMBER** {units=**NX_LENGTH**}

opening along the *x* axis

**y_gap**: (optional) **NX_NUMBER** {units=**NX_LENGTH**}

opening along the *y* axis

**COLLIMATOR**: (optional) **NXcollimator**

Description of a collimating element (defines the divergence of the beam) in the instrument.

To document a slit, pinhole, or the beam, refer to the documentation of the NXinstrument group above.

@canSAS_class: (required) **NX_CHAR**

Official canSAS group: NXcanSAS (applications); SAScollimation

Obligatory value: SAScollimation

**length**: (optional) **NX_NUMBER** {units=**NX_LENGTH**}

Amount/length of collimation inserted (as on a SANS instrument)

**distance**: (optional) **NX_NUMBER** {units=**NX_LENGTH**}

Distance from this collimation element to the sample

**DETECTOR**: (optional) **NXdetector**

Description of a detector in the instrument.

@canSAS_class: (required) **NX_CHAR**

Official canSAS group: NXcanSAS (applications); SASdetector

Obligatory value: SASdetector

**name**: (required) **NX_CHAR**

Identifies the name of this detector

**SDD**: (optional) **NX_NUMBER** {units=**NX_LENGTH**}

Distance between sample and detector.

Note: In NXdetector, the distance field records the distance to the previous component … most often the sample. This use is the same
as SDD for most SAS instruments but not all. For example, Bonse-Hart cameras have one or more crystals between the sample and detector.

We define here the field SDD to document without ambiguity the distance between sample and detector.

**slit_length**: (optional) \texttt{NX_NUMBER \{units=NX\_PER\_LENGTH\}}

Slit length of the instrument for this detector, expressed in the same units as \(Q\).

**x\_position**: (optional) \texttt{NX_NUMBER \{units=NX\_LENGTH\}}

Location of the detector in \(x\)

**y\_position**: (optional) \texttt{NX_NUMBER \{units=NX\_LENGTH\}}

Location of the detector in \(y\)

**roll**: (optional) \texttt{NX_NUMBER \{units=NX\_ANGLE\}}

Rotation of the detector about the \(z\) axis (roll)

**pitch**: (optional) \texttt{NX_NUMBER \{units=NX\_ANGLE\}}

Rotation of the detector about the \(x\) axis (roll)

**yaw**: (optional) \texttt{NX_NUMBER \{units=NX\_ANGLE\}}

Rotation of the detector about the \(y\) axis (yaw)

**beam\_center\_x**: (optional) \texttt{NX\_FLOAT \{units=NX\_LENGTH\}}

Position of the beam center on the detector.

This is the \(x\) position where the direct beam would hit the detector plane. This is a length and can be outside of the actual detector. The length can be in physical units or pixels as documented by the units attribute. The value can be any real number (positive, zero, or negative).

**beam\_center\_y**: (optional) \texttt{NX\_FLOAT \{units=NX\_LENGTH\}}

Position of the beam center on the detector.

This is the \(y\) position where the direct beam would hit the detector plane. This is a length and can be outside of the actual detector. The length can be in physical units or pixels as documented by the units attribute. The value can be any real number (positive, zero, or negative).

**x\_pixel\_size**: (optional) \texttt{NX\_FLOAT \{units=NX\_LENGTH\}}

Size of each detector pixel. If it is scalar all pixels are the same size

**y\_pixel\_size**: (optional) \texttt{NX\_FLOAT \{units=NX\_LENGTH\}}

Size of each detector pixel. If it is scalar all pixels are the same size

**SOURCE**: (optional) \texttt{NXsource}

Description of the radiation source.

@\texttt{canSAS\_class}: (required) \texttt{NX\_CHAR}

Official canSAS group: NXcanSAS (applications); SASsource

Obligatory value: SASsource

**radiation**: (optional) \texttt{NX\_CHAR}
**DEPRECATED**: Use either (or both) `probe` or `type` fields from `NXsource` (issue #765)

Name of the radiation used. Note that this is **not** the name of the facility!

This field contains a value from either the `probe` or `type` fields in `NXsource`. Thus, it is redundant with existing NeXus structure.

Any of these values:
- 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
- Ion Source
- UV Plasma Source
- neutron
- x-ray
- muon
- electron
- ultraviolet
- visible light
- positron
- proton

**beam_shape**: (optional) `NX_CHAR`

Text description of the shape of the beam (incident on the sample).

**incident_wavelength**: (optional) `NX_NUMBER {units=NX_WAVELENGTH}`

Wavelength ($\lambda$) of radiation incident on the sample.

**wavelength_min**: (optional) `NX_NUMBER {units=NX_WAVELENGTH}`

Some facilities specify wavelength using a range. This is the lowest wavelength in such a range.

**wavelength_max**: (optional) `NX_NUMBER {units=NX_WAVELENGTH}`

Some facilities specify wavelength using a range. This is the highest wavelength in such a range.
incident_wavelength_spread: (optional) NX_NUMBER {units=NX_WAVELENGTH}

Some facilities specify wavelength using a range. This is the width (FWHM) of such a range.

beam_size_x: (optional) NX_NUMBER {units=NX_LENGTH}

Size of the incident beam along the x axis.

beam_size_y: (optional) NX_NUMBER {units=NX_LENGTH}

Size of the incident beam along the y axis.

SAMPLE: (optional) NXsample

Description of the sample.

@canSAS_class: (required) NX_CHAR

Official canSAS group: NXcanSAS (applications); SASsample

Obligatory value: SASsample

name: (required) NX_CHAR

ID: Text string that identifies this sample.

thickness: (optional) NX_FLOAT {units=NX_LENGTH}

Thickness of this sample

transmission: (optional) NX_NUMBER {units=NX_DIMENSIONLESS}

Transmission (I/I_0) of this sample. There is no units attribute as this number is dimensionless.

Note: the ability to store a transmission spectrum, instead of a single value, is provided elsewhere in the structure, in the SAStransmission_spectrum element.

temperature: (optional) NX_NUMBER {units=NX_TEMPERATURE}

Temperature of this sample.

details: (optional) NX_CHAR

Any additional sample details.

x_position: (optional) NX_NUMBER {units=NX_LENGTH}

Location of the sample in x

y_position: (optional) NX_NUMBER {units=NX_LENGTH}

Location of the sample in y

roll: (optional) NX_NUMBER {units=NX_ANGLE}

Rotation of the sample about the z axis (roll)

pitch: (optional) NX_NUMBER {units=NX_ANGLE}

Rotation of the sample about the x axis (roll)

yaw: (optional) NX_NUMBER {units=NX_ANGLE}

Rotation of the sample about the y axis (yaw)

PROCESS: (optional) NXprocess
Description of a processing or analysis step.

Add additional fields as needed to describe value(s) of any variable, parameter, or term related to the SASprocess step. Be sure to include units attributes for all numerical fields.

@canSAS_class: (required) NX_CHAR

Official canSAS group: NXcanSAS (applications); SASprocess

Obligatory value: SASprocess

term: (optional) NX_CHAR

Specifies the value of a single variable, parameter, or term (while defined here as a string, it could be a number) related to the SASprocess step.

Note: The name term is not required, it could take any name, as long as the name is unique within this group.

NOTE: (optional) NXnote

Any additional notes or subprocessing steps will be documented here.

An NXnote group can be added to any NeXus group at or below the NXentry group. It is shown here as a suggestion of a good place to consider its use.

COLLECTION: (optional) NXcollection

Describes anything about SASprocess that is not already described.

Any content not defined in the canSAS standard can be placed at this point.

Note: The name of this group is flexible, it could take any name, as long as it is unique within the NXprocess group.

@canSAS_class: (required) NX_CHAR

Official canSAS group: NXcanSAS (applications); SASprocessnote

Obligatory value: SASprocessnote

COLLECTION: (optional) NXcollection

Free form description of anything not covered by other elements.

@canSAS_class: (required) NX_CHAR

Official canSAS group: NXcanSAS (applications); SASnote

Obligatory value: SASnote

TRANSMISSION_SPECTRUM: (optional) NXdata
The \textit{SAstransmission\_spectrum} element

This describes certain data obtained from a variable-wavelength source such as pulsed-neutron source.

The name of each \textit{SAstransmission\_spectrum} group must be unique within a \textit{SASentry} group. Suggest using names such as \textit{sastransmission\_spectrum01}.

\texttt{@canSAS\_class}: (required) \texttt{NX\_CHAR}

Official canSAS group: NXcanSAS (applications); SAstransmission\_spectrum

Obligatory value: \textit{SAstransmission\_spectrum}

\texttt{@signal}: (required) \texttt{NX\_CHAR}

Name of the default data field.

Obligatory value:

- \texttt{T}: For \textit{SAstransmission\_spectrum}, this is always “\texttt{T}”.

\texttt{@T\_axes}: (required) \texttt{NX\_CHAR}

Obligatory value:

- \texttt{T}: the wavelengths field (as a dimension scale) corresponding to this transmission

\texttt{@name}: (required) \texttt{NX\_CHAR}

Identify what type of spectrum is being described. It is expected that this value will take either of these two values:

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample</td>
<td>measurement with the sample and container</td>
</tr>
<tr>
<td>can</td>
<td>measurement with just the container</td>
</tr>
</tbody>
</table>

\texttt{@timestamp}: (optional) \texttt{NX\_DATE\_TIME}

ISO-8601 time

\texttt{lambda}: (required) \texttt{NX\_NUMBER} \{units=\texttt{NX\_WAVELENGTH} \}

Wavelength of the radiation.

This array is of the same shape as \texttt{T} and \texttt{Tdev}.

\texttt{T}: (required) \texttt{NX\_NUMBER} \{units=\texttt{NX\_DIMENSIONLESS} \}

Transmission values ($I/I_0$) as a function of wavelength.

This array is of the same shape as \texttt{lambda} and \texttt{Tdev}.

\texttt{@uncertainties}: (required) \texttt{NX\_CHAR}

Names the dataset (in this SASdata group) that provides the uncertainty of each transmission \texttt{T} to be used for data analysis. The name of the dataset containing the \texttt{T} uncertainty is expected to be \texttt{Tdev}.

Typically:

\texttt{@uncertainties=“Tdev”}

\texttt{Tdev}: (required) \texttt{NX\_NUMBER} \{units=\texttt{NX\_DIMENSIONLESS} \}
Estimated uncertainty (usually standard deviation) in $T$. Must have the same units as $T$.

This is the field is named in the `uncertainties` attribute of $T$, as in:

$$T/@uncertainties="Tdev"$$

This array is of the same shape as $\lambda$ and $T$.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXcanSAS/ENTRY-group`
- `/NXcanSAS/ENTRY/COLLECTION-group`
- `/NXcanSAS/ENTRY/COLLECTION@canSAS_class-attribute`
- `/NXcanSAS/ENTRY/DATA-group`
- `/NXcanSAS/ENTRY/DATA/dQl-field`
- `/NXcanSAS/ENTRY/DATA/dQl@units-attribute`
- `/NXcanSAS/ENTRY/DATA/dQw-field`
- `/NXcanSAS/ENTRY/DATA/dQw@units-attribute`
- `/NXcanSAS/ENTRY/DATA/I-field`
- `/NXcanSAS/ENTRY/DATA/I@scaling_factor-attribute`
- `/NXcanSAS/ENTRY/DATA/I@uncertainties-attribute`
- `/NXcanSAS/ENTRY/DATA/I@units-attribute`
- `/NXcanSAS/ENTRY/DATA/Idev-field`
- `/NXcanSAS/ENTRY/DATA/Idev@units-attribute`
- `/NXcanSAS/ENTRY/DATA/Q-field`
- `/NXcanSAS/ENTRY/DATA/Q@resolutions-attribute`
- `/NXcanSAS/ENTRY/DATA/Q@resolutions_description-attribute`
- `/NXcanSAS/ENTRY/DATA/Q@uncertainties-attribute`
- `/NXcanSAS/ENTRY/DATA/Q@units-attribute`
- `/NXcanSAS/ENTRY/DATA/Qdev-field`
- `/NXcanSAS/ENTRY/DATA/Qdev@units-attribute`
- `/NXcanSAS/ENTRY/DATA/Qmean-field`
- `/NXcanSAS/ENTRY/DATA/Qmean@units-attribute`
- `/NXcanSAS/ENTRY/DATA/ShadowFactor-field`
- `/NXcanSAS/ENTRY/DATA@canSAS_class-attribute`
- `/NXcanSAS/ENTRY/DATA@I_axes-attribute`
- `/NXcanSAS/ENTRY/DATA@mask-attribute`
- `/NXcanSAS/ENTRY/DATA@Mask_indices-attribute`
• /NXcanSAS/ENTRY/DATA/Q_indices-attribute
• /NXcanSAS/ENTRY/DATA/@signal-attribute
• /NXcanSAS/ENTRY/DATA/@timestamp-attribute
• /NXcanSAS/ENTRY/definition-field
• /NXcanSAS/ENTRY/INSTRUMENT-group
• /NXcanSAS/ENTRY/INSTRUMENT/APERTURE-group
  • /NXcanSAS/ENTRY/INSTRUMENT/APERTURE/shape-field
  • /NXcanSAS/ENTRY/INSTRUMENT/APERTURE/x_gap-field
  • /NXcanSAS/ENTRY/INSTRUMENT/APERTURE/y_gap-field
• /NXcanSAS/ENTRY/INSTRUMENT/APERTURE/@canSAS_class-attribute
• /NXcanSAS/ENTRY/INSTRUMENT/COLLIMATOR-group
  • /NXcanSAS/ENTRY/INSTRUMENT/COLLIMATOR/distance-field
  • /NXcanSAS/ENTRY/INSTRUMENT/COLLIMATOR/length-field
• /NXcanSAS/ENTRY/INSTRUMENT/COLLIMATOR/@canSAS_class-attribute
• /NXcanSAS/ENTRY/INSTRUMENT/DETECTOR-group
  • /NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/beam_center_x-field
  • /NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/beam_center_y-field
• /NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/name-field
• /NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/pitch-field
• /NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/roll-field
• /NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/SDD-field
• /NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/slit_length-field
• /NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/x_pixel_size-field
• /NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/x_position-field
• /NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/y_pixel_size-field
• /NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/y_position-field
• /NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/yaw-field
• /NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/@canSAS_class-attribute
• /NXcanSAS/ENTRY/INSTRUMENT/SOURCE-group
• /NXcanSAS/ENTRY/INSTRUMENT/SOURCE/beam_shape-field
• /NXcanSAS/ENTRY/INSTRUMENT/SOURCE/beam_size_x-field
• /NXcanSAS/ENTRY/INSTRUMENT/SOURCE/beam_size_y-field
• /NXcanSAS/ENTRY/INSTRUMENT/SOURCE/incident_wavelength-field
• /NXcanSAS/ENTRY/INSTRUMENT/SOURCE/incident_wavelength_spread-field
• /NXcanSAS/ENTRY/INSTRUMENT/SOURCE/radiation-field
• /NXcanSAS/ENTRY/INSTRUMENT/SOURCE/wavelength_max-field
3.3. NeXus Class Definitions

- /NXcanSAS/ENTRY/INSTRUMENT/SOURCE/wavelength_min-field
- /NXcanSAS/ENTRY/INSTRUMENT/SOURCE@canSAS_class-attribute
- /NXcanSAS/ENTRY/INSTRUMENT@canSAS_class-attribute
- /NXcanSAS/ENTRY/PROCESS-group
- /NXcanSAS/ENTRY/PROCESS/COLLECTION-group
- /NXcanSAS/ENTRY/PROCESS/COLLECTION@canSAS_class-attribute
- /NXcanSAS/ENTRY/PROCESS/date-field
- /NXcanSAS/ENTRY/PROCESS/description-field
- /NXcanSAS/ENTRY/PROCESS/name-field
- /NXcanSAS/ENTRY/PROCESS/NOTE-group
- /NXcanSAS/ENTRY/PROCESS/term-field
- /NXcanSAS/ENTRY/PROCESS@canSAS_class-attribute
- /NXcanSAS/ENTRY/run-field
- /NXcanSAS/ENTRY/run@name-attribute
- /NXcanSAS/ENTRY/SAMPLE-group
- /NXcanSAS/ENTRY/SAMPLE/details-field
- /NXcanSAS/ENTRY/SAMPLE/name-field
- /NXcanSAS/ENTRY/SAMPLE/pitch-field
- /NXcanSAS/ENTRY/SAMPLE/roll-field
- /NXcanSAS/ENTRY/SAMPLE/temperature-field
- /NXcanSAS/ENTRY/SAMPLE/thickness-field
- /NXcanSAS/ENTRY/SAMPLE/transmission-field
- /NXcanSAS/ENTRY/SAMPLE/x_position-field
- /NXcanSAS/ENTRY/SAMPLE/y_position-field
- /NXcanSAS/ENTRY/SAMPLE/yaw-field
- /NXcanSAS/ENTRY/SAMPLE@canSAS_class-attribute
- /NXcanSAS/ENTRY/title-field
- /NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM-group
- /NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM/lambda-field
- /NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM/T-field
- /NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM/T@uncertainties-attribute
- /NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM/Tdev-field
- /NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM@canSAS_class-attribute
- /NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM@name-attribute
- /NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM@signal-attribute
- /NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM@T_axes-attribute
NXdirecttof

Status:
application definition, extends NXtofraw

Description:
This is a application definition for raw data from a direct geometry TOF spectrometer

Symbols:
No symbol table

Groups cited:
NXdisk_chopper, NXentry, NXfermi_chopper, NXinstrument

Structure:

entry: (required) NXentry
  
  title: (required) NX_CHAR
  
  start_time: (required) NX_DATE_TIME
  
  definition: (required) NX_CHAR

  Official NeXus NXDL schema to which this file conforms

  Obligatory value: NXdirecttof

INSTRUMENT: (required) NXinstrument

  We definitely want the rotation_speed and energy of the chopper. Thus either a
  fermi_chopper or a disk_chopper group is required.

  fermi_chopper: (optional) NXfermi_chopper

  rotation_speed: (required) NX_FLOAT {units=NX_FREQUENCY}
  
  chopper rotation speed

  energy: (required) NX_FLOAT {units=NX ENERGY}
  
  energy selected

  disk_chopper: (optional) NXdisk_chopper

  rotation_speed: (required) NX_FLOAT {units=NX_FREQUENCY}
  
  chopper rotation speed

  energy: (required) NX_FLOAT {units=NX ENERGY}
  
  energy selected
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXdirectof/entry-group
- /NXdirectof/entry/definition-field
- /NXdirectof/entry/INSTRUMENT-group
- /NXdirectof/entry/INSTRUMENT/disk_chopper-group
- /NXdirectof/entry/INSTRUMENT/disk_chopper/energy-field
- /NXdirectof/entry/INSTRUMENT/disk_chopper/rotation_speed-field
- /NXdirectof/entry/INSTRUMENT/fermi_chopper-group
- /NXdirectof/entry/INSTRUMENT/fermi_chopper/energy-field
- /NXdirectof/entry/INSTRUMENT/fermi_chopper/rotation_speed-field
- /NXdirectof/entry/start_time-field
- /NXdirectof/entry/title-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXdirectof.nxdl.xml

NXfluo

Status:
application definition, extends NXobject

Description:
This is an application definition for raw data from an X-ray fluorescence experiment

Symbols:
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nE: Number of energies

Groups cited:
NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXmonochromator, NXsample, NXsource

Structure:

entry: (required) NXentry
  title: (required) NX_CHAR
  start_time: (required) NX_DATE_TIME
  definition: (required) NX_CHAR
    Official NeXus NXDL schema to which this file conforms.
    Obligatory value: NXfluo

INSTRUMENT: (required) NXinstrument

SOURCE: (required) NXsource
**type**: (required) \textit{NX\_CHAR}

**name**: (required) \textit{NX\_CHAR}

**probe**: (required) \textit{NX\_CHAR}

Obligatory value: \textit{x-ray}

**monochromator**: (required) \textit{NXmonochromator}

**wavelength**: (required) \textit{NX\_FLOAT}

**fluorescence**: (required) \textit{NXdetector}

**data**: (required) \textit{NX\_INT} (Rank: 1, Dimensions: [nE])

**energy**: (required) \textit{NX\_FLOAT} (Rank: 1, Dimensions: [nE])

**SAMPLE**: (required) \textit{NXsample}

**name**: (required) \textit{NX\_CHAR}

Descriptive name of sample

**MONITOR**: (required) \textit{NXmonitor}

**mode**: (required) \textit{NX\_CHAR}

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: \textit{monitor} | \textit{timer}

**preset**: (required) \textit{NX\_FLOAT}

preset value for time or monitor

**data**: (required) \textit{NX\_INT}

**data**: (required) \textit{NXdata}

**energy**: \textit{link} (suggested target: /entry/instrument/fluorescence/energy)

**data**: \textit{link} (suggested target: /entry/instrument/fluorescence/data)

### Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXfluo/entry-group
- /NXfluo/entry/data-group
- /NXfluo/entry/data/data-link
- /NXfluo/entry/data/energy-link
- /NXfluo/entry/definition-field
- /NXfluo/entry/INSTRUMENT-group
- /NXfluo/entry/INSTRUMENT/fluorescence-group
- /NXfluo/entry/INSTRUMENT/fluorescence/data-field
- /NXfluo/entry/INSTRUMENT/fluorescence/energy-field
- /NXfluo/entry/INSTRUMENT/monochromator-group
• /NXfluo/entry/INSTRUMENT/monochromator/wavelength-field
• /NXfluo/entry/INSTRUMENT/SOURCE-group
• /NXfluo/entry/INSTRUMENT/SOURCE/name-field
• /NXfluo/entry/INSTRUMENT/SOURCE/probe-field
• /NXfluo/entry/INSTRUMENT/SOURCE/type-field
• /NXfluo/entry/MONITOR-group
• /NXfluo/entry/MONITOR/data-field
• /NXfluo/entry/MONITOR/mode-field
• /NXfluo/entry/MONITOR/preset-field
• /NXfluo/entry/SAMPLE-group
• /NXfluo/entry/SAMPLE/name-field
• /NXfluo/entry/start_time-field
• /NXfluo/entry/title-field

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/applications/NXfluo.nxdl.xml

**NXindirecttof**

**Status:**
application definition, extends **NXtofraw**

**Description:**
This is a application definition for raw data from a direct geometry TOF spectrometer

**Symbols:**
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

**nDet**: Number of detectors

**Groups cited:**
NXentry, NXinstrument, NXmonochromator

**Structure:**

.entry: (required) **NXentry**
  .title: (required) **NX_CHAR**
  .start_time: (required) **NX_DATE_TIME**
  .definition: (required) **NX_CHAR**
    Official NeXus NXDL schema to which this file conforms
    Obligatory value: **NXindirecttof**

  .INSTRUMENT: (required) **NXinstrument**
    .analyser: (required) **NXmonochromator**
      .energy: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nDet])
        {units=**NX_ENERGY**}
analyzed energy

**polar_angle:** (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet])
{units=*NX_ANGLE*}

  polar angle towards sample

**distance:** (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet])
{units=*NX_LENGTH*}

  distance from sample

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXindirecttof/entry-group
- /NXindirecttof/entry/definition-field
- /NXindirecttof/entry/INSTRUMENT-group
- /NXindirecttof/entry/INSTRUMENT/analyser-group
- /NXindirecttof/entry/INSTRUMENT/analyser/distance-field
- /NXindirecttof/entry/INSTRUMENT/analyser/energy-field
- /NXindirecttof/entry/INSTRUMENT/analyser/polar_angle-field
- /NXindirecttof/entry/start_time-field
- /NXindirecttof/entry/title-field

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/applications/NXindirecttof.nxdl.xml

**NXiqproc**

**Status:**

  application definition, extends *NXobject*

**Description:**

  Application definition for any *I(Q)* data.

**Symbols:**

  The symbol(s) listed here will be used below to coordinate datasets with the same shape.

  - **nVars:** The number of values taken by the varied variable
  - **nQX:** Number of values for the first dimension of Q
  - **nQY:** Number of values for the second dimension of Q

**Groups cited:**

  *NXdata, NXentry, NXinstrument, NXparameters, NXprocess, NXsample, NXsource*

**Structure:**

  **ENTRY:** (required) *NXentry*
@entry: (required) **NX_CHAR**

title: (required) **NX_CHAR**

definition: (required) **NX_CHAR**

   Official NeXus NXDL schema to which this file conforms

   Obligatory value: @entry

instrument: (required) **NXinstrument**

   name: (required) **NX_CHAR**

   Name of the instrument from which this data was reduced.

SOURCE: (required) **NXsource**

type: (required) **NX_CHAR**

name: (required) **NX_CHAR**

probe: (required) **NX_CHAR**

   Any of these values: neutron | x-ray | electron

SAMPLE: (required) **NXsample**

name: (required) **NX_CHAR**

   Descriptive name of sample

reduction: (required) **NXprocess**

program: (required) **NX_CHAR**

version: (required) **NX_CHAR**

input: (required) **NXparameters**

   Input parameters for the reduction program used

filenames: (required) **NX_CHAR**

   Raw data files used to generate this I(Q)

output: (required) **NXparameters**

   Eventual output parameters from the data reduction program used

DATA: (required) **NXdata**

data: (required) **NX_INT** (Rank: 3, Dimensions: [nVars, nQX, nQY])

   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

variable: (required) **NX_NUMBER** (Rank: 1, Dimensions: [nVars])

   @varied_variable: (required) **NX_CHAR**

   The real name of the varied variable in the first dim of data, temperature, P, MF etc…

qx: (required) **NX_NUMBER** (Rank: 1, Dimensions: [nQX])

   Values for the first dimension of Q

3.3. NeXus Class Definitions
**qy**: (required) *NX_NUMBER* (Rank: 1, Dimensions: [nQY])
Values for the second dimension of Q

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXiqproc/ENTRY-group
- /NXiqproc/ENTRY/DATA-group
- /NXiqproc/ENTRY/DATA/data-field
- /NXiqproc/ENTRY/DATA/qx-field
- /NXiqproc/ENTRY/DATA/qy-field
- /NXiqproc/ENTRY/DATA/variable-field
- /NXiqproc/ENTRY/DATA/variable@varied_variable-attribute
- /NXiqproc/ENTRY/definition-field
- /NXiqproc/ENTRY/instrument-group
- /NXiqproc/ENTRY/instrument/name-field
- /NXiqproc/ENTRY/instrument/SOURCE-group
- /NXiqproc/ENTRY/instrument/SOURCE/name-field
- /NXiqproc/ENTRY/instrument/SOURCE/probe-field
- /NXiqproc/ENTRY/instrument/SOURCE/type-field
- /NXiqproc/ENTRY/reduction-group
- /NXiqproc/ENTRY/reduction/input-group
- /NXiqproc/ENTRY/reduction/input/filenames-field
- /NXiqproc/ENTRY/reduction/output-group
- /NXiqproc/ENTRY/reduction/program-field
- /NXiqproc/ENTRY/reduction/version-field
- /NXiqproc/ENTRY/SAMPLE-group
- /NXiqproc/ENTRY/SAMPLE/name-field
- /NXiqproc/ENTRY/title-field
- /NXiqproc/ENTRY@entry-attribute

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/applications/NXiqproc.nxdl.xml
NXlauetof

Status:

  application definition, extends NXobject

Description:

  This is the application definition for a TOF laue diffractometer

Symbols:

  The symbol(s) listed here will be used below to coordinate datasets with the same shape.

  nXPixels: Number of X pixels
  nYPixels: Number of Y pixels
  nTOF: Time of flight

Groups cited:

  NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample

Structure:

  entry: (required) NXentry
    
    definition: (required) NX_CHAR
      
      Official NeXus NXDL schema to which this file conforms
      
      Obligatory value: NXlauetof

    instrument: (required) NXinstrument
      
      detector: (required) NXdetector
        
        This assumes a planar 2D detector. All angles and distances refer to the center of the detector.

        polar_angle: (required) NX_FLOAT {units=NX_ANGLE}
          
          The polar_angle (two theta) where the detector is placed.

        azimuthal_angle: (required) NX_FLOAT {units=NX_ANGLE}
          
          The azimuthal angle where the detector is placed.

        data: (required) NX_INT (Rank: 3, Dimensions: [nXPixels, nYPixels, nTOF])

          @signal: (required) NX_POSINT
            
            Obligatory value: 1

        x_pixel_size: (required) NX_FLOAT {units=NX_LENGTH}

        y_pixel_size: (required) NX_FLOAT {units=NX_LENGTH}

        distance: (required) NX_FLOAT {units=NX_LENGTH}

        time_of_flight: (required) NX_FLOAT (Rank: 1, Dimensions: [nTOF])
          
          {units=NX_TIME_OF_FLIGHT}

    sample: (required) NXsample
      
      name: (required) NX_CHAR
        
        Descriptive name of sample
**orientation_matrix**: (required) `NX_FLOAT` (Rank: 2, Dimensions: [3, 3])

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 the UB nearly always.

**unit_cell**: (required) `NX_FLOAT` (Rank: 1, Dimensions: [6])

The unit cell, a, b, c, alpha, beta, gamma. Again, not strictly necessary, but normally written.

**control**: (required) `NXmonitor`

**mode**: (required) `NX_CHAR`

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: monitor | timer

**preset**: (required) `NX_FLOAT`

preset value for time or monitor

**data**: (required) `NX_INT` (Rank: 1, Dimensions: [nTOF])

use these attributes primary=1 signal=1

**time_of_flight**: (required) `NX_FLOAT` (Rank: 1, Dimensions: [nTOF])

{units=`NX_TIME_OF_FLIGHT`}

**name**: (required) `NXdata`

**data**: link (suggested target: /NXentry/NXinstrument/NXdetector/data)

**time_of_flight**: link (suggested target: /NXentry/NXinstrument/NXdetector/time_of_flight)

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXlauetof/entry-group
- /NXlauetof/entry/control-group
- /NXlauetof/entry/control/data-field
- /NXlauetof/entry/control/mode-field
- /NXlauetof/entry/control/preset-field
- /NXlauetof/entry/control/time_of_flight-field
- /NXlauetof/entry/definition-field
- /NXlauetof/entry/instrument-group
- /NXlauetof/entry/instrument/detector-group
- /NXlauetof/entry/instrument/detector/azimuthal_angle-field
- /NXlauetof/entry/instrument/detector/data-field
- /NXlauetof/entry/instrument/detector/data@signal-attribute
NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXlauetof.nxdl.xml

**NXmonopd**

**Status:**
application definition, extends `NXobject`

**Description:**
Monochromatic Neutron and X-Ray Powder diffractometer

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.

**Symbols:**
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

- `i`: `i` is the number of wavelengths
- `nDet`: Number of detectors

**Groups cited:**
`NXcrystal, NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample, NXsource`

**Structure:**

```xml
entry: (required) NXentry

title: (required) NX_CHAR

start_time: (required) NX_DATE_TIME

definition: (required) NX_CHAR

Official NeXus NDL schema to which this file conforms

Obligatory value: NXmonopd

INSTRUMENT: (required) NXinstrument
```
SOURCE: (required) *NXsource*

  type: (required) *NX_CHAR*
  name: (required) *NX_CHAR*
  probe: (required) *NX_CHAR*

  Any of these values: neutron | x-ray | electron

CRYSTAL: (required) *NXcrystal*

  wavelength: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=NX_WAVELENGTH}

  Optimum diffracted wavelength

DETECTOR: (required) *NXdetector*

  polar_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet])
  data: (required) *NX_INT* (Rank: 1, Dimensions: [nDet])

  detector signal (usually counts) are already corrected for detector efficiency

SAMPLE: (required) *NXsample*

  name: (required) *NX_CHAR*

  Descriptive name of sample

  rotation_angle: (required) *NX_FLOAT* {units=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

MONITOR: (required) *NXmonitor*

  mode: (required) *NX_CHAR*

  Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

  Any of these values: monitor | timer

  preset: (required) *NX_FLOAT*

  preset value for time or monitor

  integral: (required) *NX_FLOAT* {units=NX_ANY}

  Total integral monitor counts

DATA: (required) *NXdata*

  polar_angle: *link* (suggested target: /NXentry/NXinstrument/NXdetector/
  polar_angle)

  Link to polar angle in /NXentry/NXinstrument/NXdetector

  data: *link* (suggested target: /NXentry/NXinstrument/NXdetector/data)

  Link to data in /NXentry/NXinstrument/NXdetector
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXmonopd/entry-group
- /NXmonopd/entry/DATA-group
- /NXmonopd/entry/DATA/data-link
- /NXmonopd/entry/DATA/polar_angle-link
- /NXmonopd/entry/definition-field
- /NXmonopd/entry/INSTRUMENT-group
- /NXmonopd/entry/INSTRUMENT/CRYSTAL-group
- /NXmonopd/entry/INSTRUMENT/CRYSTAL/wavelength-field
- /NXmonopd/entry/INSTRUMENT/DETECTOR-group
- /NXmonopd/entry/INSTRUMENT/DETECTOR/data-field
- /NXmonopd/entry/INSTRUMENT/DETECTOR/polar_angle-field
- /NXmonopd/entry/INSTRUMENT/SOURCE-group
- /NXmonopd/entry/INSTRUMENT/SOURCE/name-field
- /NXmonopd/entry/INSTRUMENT/SOURCE/probe-field
- /NXmonopd/entry/INSTRUMENT/SOURCE/type-field
- /NXmonopd/entry/MONITOR-group
- /NXmonopd/entry/MONITOR/integral-field
- /NXmonopd/entry/MONITOR/mode-field
- /NXmonopd/entry/MONITOR/preset-field
- /NXmonopd/entry/SAMPLE-group
- /NXmonopd/entry/SAMPLE/name-field
- /NXmonopd/entry/SAMPLE/rotation_angle-field
- /NXmonopd/entry/start_time-field
- /NXmonopd/entry/title-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXmonopd.nxdl.xml

NXmx

Status:
application definition, extends NXobject

Description:
functional application definition for macromolecular crystallography

Symbols:
These symbols will be used below to coordinate datasets with the same shape. Most MX x-ray detectors will produce two-dimensional images. Some will produce three-dimensional images, using one of the indices to select a detector module.

**dataRank**: Rank of the data field

**nP**: Number of scan points

**i**: Number of detector pixels in the slowest direction

**j**: Number of detector pixels in the second slowest direction

**k**: Number of detector pixels in the third slowest direction

**m**: Number of channels in the incident beam spectrum, if known

**groupIndex**: An array of the hierarchical levels of the parents of detector elements or groupings of detector elements. A top-level element or grouping has parent level -1.

**Groups cited:**

NXattenuator, NXbeam, NXcollection, NXdata, NXdetector_group, NXdetector_module, NXdetector, NXentry, NXinstrument, NXsample, NXsource, NXtransformations

**Structure:**

**ENTRY**: (required) **NXentry**

Note, it is recommended that file_name and file_time are included as attributes at the root of a file that includes NXmx. See **NXroot**.

**@version**: (optional) **NX_CHAR**

Describes the version of the NXmx definition used to write this data. This must be a text (not numerical) representation. Such as:

```
@version="1.0"
```

Obligatory value: 1.0

**title**: (optional) **NX_CHAR**

**start_time**: (required) **NX_DATE_TIME**

ISO 8601 time/date of the first data point collected in UTC, using the Z suffix to avoid confusion with local time. Note that the time zone of the beamline should be provided in NXentry/NXinstrument/time_zone.

**end_time**: (optional) **NX_DATE_TIME**

ISO 8601 time/date of the last data point collected in UTC, using the Z suffix to avoid confusion with local time. Note that the time zone of the beamline should be provided in NXentry/NXinstrument/time_zone. This field should only be filled when the value is accurately observed. If the data collection aborts or otherwise prevents accurate recording of the end_time, this field should be omitted.

**end_time_estimated**: (required) **NX_DATE_TIME**

ISO 8601 time/date of the last data point collected in UTC, using the Z suffix to avoid confusion with local time. Note that the time zone of the beamline should be provided in NXentry/NXinstrument/time_zone. This field may be filled with a value estimated before an observed value is available.

**definition**: (required) **NX_CHAR**
NeXus NXDL schema to which this file conforms

Obligatory value: NXmx

**DATA:** (required) NXdata

  **data:** (recommended) NX_NUMBER (Rank: dataRank, Dimensions: \([nP, i, j, [k]]\))

  For a dimension-2 detector, the rank of the data array will be 3. For a
dimension-3 detector, the rank of the data array will be 4. This allows for
the introduction of the frame number as the first index.

**SAMPLE:** (required) NXsample

  **name:** (required) NX_CHAR

  Descriptive name of sample

  **depends_on:** (required) NX_CHAR

  This is a requirement to describe for any scan experiment.

  The axis on which the sample position depends may be stored anywhere,
  but is normally stored in the NXtransformations group within the NXsample
  group.

  If there is no goniometer, e.g. with a jet, depends_on should be set to “.”

  **temperature:** (optional) NX_NUMBER \{units=NX_TEMPERATURE\}

**TRANSFORMATIONS:** (optional) NXtransformations

  This is the recommended location for sample goniometer and other related
  axes.

  This is a requirement to describe for any scan experiment. The reason it is
  optional is mainly to accommodate XFEL single shot exposures.

  Use of the depends_on field and the NXtransformations group is strongly
  recommended. As noted above this should be an absolute requirement to
  have for any scan experiment.

  The reason it is optional is mainly to accommodate XFEL single shot expo-
  sures.

**INSTRUMENT:** (required) NXinstrument

  **name:** (required) NX_CHAR

  Name of instrument. Consistency with the controlled vocabulary beam-
  line naming in https://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v50.dic/Items/_diffrn_source.pdbx_synchrotron_beamline.html and
  https://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v50.dic/Items/_diffrn_source.type.html is highly recommended.

  **@short_name:** (optional) NX_CHAR

  Short name for instrument, perhaps the acronym.

  **time_zone:** (recommended) NX_DATE_TIME

  ISO 8601 time_zone offset from UTC.

**ATTENUATOR:** (optional) NXattenuator

  **attenuator_transmission:** (optional) NX_NUMBER

  \{units=NX_UNITLESS\}
**DETECTOR**

Optional logical grouping of detectors.

Each detector is represented as an NXdetector with its own detector data array. Each detector data array may be further decomposed into array sections by use of NXdetector_module groups. Detectors can be grouped logically together using NXdetector_group. Groups can be further grouped hierarchically in a single NXdetector_group (for example, if there are multiple detectors at an endstation or multiple endstations at a facility). Alternatively, multiple NXdetector_groups can be provided.

The groups are defined hierarchically, with names given in the group_names field, unique identifying indices given in the field group_index, and the level in the hierarchy given in the group_parent field. For example if an x-ray detector group, DET, consists of four detectors in a rectangular array:

<table>
<thead>
<tr>
<th>DTL</th>
<th>DTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLL</td>
<td>DLR</td>
</tr>
</tbody>
</table>

We could have:

```plaintext
group_names: ["DET", "DTL", "DTR", "DLL", "DLR"]
group_index: [1, 2, 3, 4, 5]
group_parent: [-1, 1, 1, 1, 1]
```

**group_names**: (required) **NX_CHAR**

An array of the names of the detectors or the names of hierarchical groupings of detectors.

**group_index**: (required) **NX_INT** (Rank: 1, Dimensions: [i])

An array of unique identifiers for detectors or groupings of detectors.

Each ID is a unique ID for the corresponding detector or group named in the field group_names. The IDs are positive integers starting with 1.

**group_parent**: (required) **NX_INT** (Rank: 1, Dimensions: [groupIndex])

An array of the hierarchical levels of the parents of detectors or groupings of detectors.

A top-level grouping has parent level -1.

**DETECTOR**: (required) **NXdetector**

Normally the detector group will have the name detector. However, in the case of multiple detectors, each detector needs a uniquely named NXdetector.

**depends_on**: (optional) **NX_CHAR**

NeXus path to the detector positioner axis that most directly supports the detector. In the case of a single-module detector, the detector axis chain may start here.

**data**: (recommended) **NX_NUMBER** (Rank: dataRank, Dimensions: [nP, i, j, k])

For a dimension-2 detector, the rank of the data array will be 3. For a dimension-3 detector, the rank of the data array will be 4. This allows for the introduction of the frame number as the first index.
**description:** (recommended) *NX_CHAR*

name/manufacturer/model/etc. information.

**time_per_channel:** (optional) *NX_CHAR {units=NX_TIME}*

For a time-of-flight detector this is the scaling factor to convert from the numeric value reported to the flight time for a given measurement.

**distance:** (recommended) *NX_FLOAT {units=NX_LENGTH}*

Distance from the sample to the beam center. Normally this value is for guidance only, the proper geometry can be found following the depends_on axis chain. But in appropriate cases where the detector distance to the sample is observable independent of the axis chain, that may take precedence over the axis chain calculation.

**distance_derived:** (recommended) *NX_BOOLEAN*

Boolean to indicate if the distance is a derived, rather than a primary observation. If distance_derived true or is not specified, the distance is assumed to be derived from detector axis specifications.

**dead_time:** (optional) *NX_FLOAT {units=NX_TIME}*

Detector dead time.

**count_time:** (recommended) *NX_NUMBER {units=NX_TIME}*

Elapsed actual counting time.

**beam_center_derived:** (optional) *NX_BOOLEAN*

Boolean to indicate if the distance is a derived, rather than a primary observation. If true or not provided, that value of beam_center_derived is assumed to be true.

**beam_center_x:** (recommended) *NX_FLOAT {units=NX_LENGTH}*

This is the x position where the direct beam would hit the detector. This is a length and can be outside of the actual detector. The length can be in physical units or pixels as documented by the units attribute. Normally, this should be derived from the axis chain, but the direct specification may take precedence if it is not a derived quantity.

**beam_center_y:** (recommended) *NX_FLOAT {units=NX_LENGTH}*

This is the y position where the direct beam would hit the detector. This is a length and can be outside of the actual detector. The length can be in physical units or pixels as documented by the units attribute. Normally, this should be derived from the axis chain, but the direct specification may take precedence if it is not a derived quantity.

**angular_calibration_applied:** (optional) *NX_BOOLEAN*

True when the angular calibration has been applied in the electronics, false otherwise.

**angular_calibration:** (optional) *NX_FLOAT (Rank: dataRank, Dimensions: [i, j, [k]])*

Angular calibration data.

**flatfield_applied:** (optional) *NX_BOOLEAN*
True when the flat field correction has been applied in the electronics, false otherwise.

**flatfield**: (optional) *NX_NUMBER* (Rank: dataRank, Dimensions: [i, j, [k]])

Flat field correction data. If provided, it is recommended that it be compressed.

**flatfield_error**: (optional) *NX_NUMBER* (Rank: dataRank, Dimensions: [i, j, [k]])

*Deprecated form. Use plural form* Errors of the flat field correction data. If provided, it is recommended that it be compressed.

**flatfield_errors**: (optional) *NX_NUMBER* (Rank: dataRank, Dimensions: [i, j, [k]])

Errors of the flat field correction data. If provided, it is recommended that it be compressed.

**pixel_mask_applied**: (optional) *NX_BOOLEAN*

True when the pixel mask correction has been applied in the electronics, false otherwise.

**pixel_mask**: (recommended) *NX_INT* (Rank: 2, Dimensions: [i, j])

The 32-bit pixel mask for the detector. Can be either one mask for the whole dataset (i.e. an array with indices i, j) or each frame can have its own mask (in which case it would be an array with indices nP, i, j).

Contains a bit field for each pixel to signal dead, blind, high or otherwise unwanted or undesirable pixels. They have the following meaning:

- bit 0: gap (pixel with no sensor)
- bit 1: dead
- bit 2: under-responding
- bit 3: over-responding
- bit 4: noisy
- bit 5: -undefined-
- bit 6: pixel is part of a cluster of problematic pixels (bit set in addition to others)
- bit 7: -undefined-
- bit 8: user defined mask (e.g. around beamstop)
- bits 9-30: -undefined-
- bit 31: virtual pixel (corner pixel with interpolated value)

Normal data analysis software would not take pixels into account when a bit in (mask & 0x0000FFFF) is set. Tag bit in the upper two bytes would indicate special pixel properties that normally would not be a sole reason to reject the intensity value (unless lower bits are set.

If the full bit depths is not required, providing a mask with fewer bits is permissible.
If needed, additional pixel masks can be specified by including additional entries named pixel_mask_N, where N is an integer. For example, a general bad pixel mask could be specified in pixel_mask that indicates noisy and dead pixels, and an additional pixel mask from experiment-specific shadowing could be specified in pixel_mask_2. The cumulative mask is the bitwise OR of pixel_mask and any pixel_mask_N entries.

If provided, it is recommended that it be compressed.

count_rate_correction_applied: (optional) NX_BOOLEAN

True when a count-rate correction has already been applied in the data recorded here, false otherwise.

virtual_pixel_interpolation_applied: (optional) NX_BOOLEAN

True when virtual pixel interpolation has been applied, false otherwise.

When virtual pixel interpolation is applied, values of some pixels may contain interpolated values. For example, to account for space between readout chips on a module, physical pixels on edges and corners between chips may have larger sensor areas and counts may be distributed between their logical pixels.

bit_depth_readout: (recommended) NX_INT

How many bits the electronics record per pixel.

detector_readout_time: (optional) NX_FLOAT {units=NX_TIME}

Time it takes to read the detector (typically milliseconds). This is important to know for time resolved experiments.

frame_time: (optional) NX_FLOAT {units=NX_TIME}

This is time for each frame. This is exposure_time + readout time.

gain_setting: (optional) NX_CHAR

The gain setting of the detector. This influences background. This is a detector-specific value meant to document the gain setting of the detector during data collection, for detectors with multiple available gain settings.

Examples of gain settings include:

- standard
- fast
- auto
- high
- medium
- low
- mixed high to medium
- mixed medium to low

Developers are encouraged to use one of these terms, or to submit additional terms to add to the list.

saturation_value: (optional) NX_INT
The value at which the detector goes into saturation. Data above this value is known to be invalid.

For example, given a saturation_value and an underload_value, the valid pixels are those less than or equal to the saturation_value and greater than or equal to the underload_value.

**underload_value**: (optional) `NX_INT`

The lowest value at which pixels for this detector would be reasonably be measured.

For example, given a saturation_value and an underload_value, the valid pixels are those less than or equal to the saturation_value and greater than or equal to the underload_value.

**sensor_material**: (required) `NX_CHAR`

At times, radiation is not directly sensed by the detector. Rather, the detector might sense the output from some converter like a scintillator. This is the name of this converter material.

**sensor_thickness**: (required) `NX_FLOAT` {units=`NX_LENGTH`}

At times, radiation is not directly sensed by the detector. Rather, the detector might sense the output from some converter like a scintillator. This is the thickness of this converter material.

**threshold_energy**: (optional) `NX_FLOAT` {units=`NX_ENERGY`}

Single photon counter detectors can be adjusted for a certain energy range in which they work optimally. This is the energy setting for this. If the detector supports multiple thresholds, this is an array.

**type**: (optional) `NX_CHAR`

Description of type such as scintillator, ccd, pixel, image plate, CMOS, ...

**TRANSFORMATIONS**: (optional) `NXtransformations`

Location for axes (transformations) to do with the detector. In the case of a single-module detector, the axes of the detector axis chain may be stored here.

**COLLECTION**: (optional) `NXcollection`

Suggested container for detailed non-standard detector information like corrections applied automatically or performance settings.

**DETECTOR_MODULE**: (required) `NXdetector_module`

Many detectors consist of multiple smaller modules that are operated in sync and store their data in a common dataset. To allow consistent parsing of the experimental geometry, this application definition requires all detectors to define a detector module, even if there is only one.

This group specifies the hyperslab of data in the data array associated with the detector that contains the data for this module. If the module is associated with a full data array, rather than with a hyperslab within a larger array, then a single module should be defined, spanning the entire array.
Note, the pixel size is given as values in the array fast_pixel_direction and slow_pixel_direction.

**data_origin**: (required) \texttt{NX_INT}

A dimension-2 or dimension-3 field which gives the indices of the origin of the hyperslab of data for this module in the main area detector image in the parent NXdetector module.

The data_origin is 0-based.

The frame number dimension (nP) is omitted. Thus the data_origin field for a dimension-2 dataset with indices (nP, i, j) will be an array with indices (i, j), and for a dimension-3 dataset with indices (nP, i, j, k) will be an array with indices (i, j, k).

The order of indices (i, j or i, j, k) is slow to fast.

**data_size**: (required) \texttt{NX_INT}

Two or three values for the size of the module in pixels in each direction. Dimensionality and order of indices is the same as for data_origin.

**data_stride**: (optional) \texttt{NX_INT}

Two or three values for the stride of the module in pixels in each direction. By default the stride is \([1,1]\) or \([1,1,1]\), and this is the most likely case. This optional field is included for completeness.

**module_offset**: (optional) \texttt{NX_NUMBER \{units=NX_LENGTH\}}

Offset of the module in regards to the origin of the detector in an arbitrary direction.

@transformation_type: (required) \texttt{NX_CHAR}

Obligatory value: translation

@vector: (required) \texttt{NX_NUMBER}

@offset: (required) \texttt{NX_NUMBER}

@depends_on: (required) \texttt{NX_CHAR}

**fast_pixel_direction**: (required) \texttt{NX_NUMBER \{units=NX_LENGTH\}}

Values along the direction of \textit{fastest varying} pixel direction. The direction itself is given through the vector attribute.

@transformation_type: (required) \texttt{NX_CHAR}

Obligatory value: translation

@vector: (required) \texttt{NX_NUMBER}

@offset: (required) \texttt{NX_NUMBER}

@depends_on: (required) \texttt{NX_CHAR}

**slow_pixel_direction**: (required) \texttt{NX_NUMBER \{units=NX_LENGTH\}}

Values along the direction of \textit{slowest varying} pixel direction. The direction itself is given through the vector attribute.
Obligatory value: translation
@vector: (required) NX_NUMBER
@offset: (required) NX_NUMBER
@depends_on: (required) NX_CHAR

BEAM: (required) NXbeam
@flux: (optional) NX_CHAR

Which field contains the measured flux. One of flux, total_flux, flux_integrated, or total_flux_integrated.

Alternatively, the name being indicated could be a link to a dataset in an NXmonitor group that records per shot beam data.

incident_wavelength: (required) NX_FLOAT {units=NX_WAVELENGTH}

In the case of a monochromatic beam this is the scalar wavelength.

Several other use cases are permitted, depending on the presence or absence of other incident_wavelength_X fields.

In the case of a polychromatic beam this is an array of length m of wavelengths, with the relative weights in incident_wavelength_weights.

In the case of a monochromatic beam that varies shot-to-shot, this is an array of wavelengths, one for each recorded shot. Here, incident_wavelength_weights and incident_wavelength_spread are not set.

In the case of a polychromatic beam that varies shot-to-shot, this is an array of length m with the relative weights in incident_wavelength_weights as a 2D array.

In the case of a polychromatic beam that varies shot-to-shot and where the channels also vary, this is a 2D array of dimensions nP by m (slow to fast) with the relative weights in incident_wavelength_weights as a 2D array.

Note, variants are a good way to represent several of these use cases in a single dataset, e.g. if a calibrated, single-value wavelength value is available along with the original spectrum from which it was calibrated.

incident_wavelength_weight: (optional) NX_FLOAT

DEPRECATED: use incident_wavelength_weights, see https://github.com/nexusformat/definitions/issues/837

In the case of a polychromatic beam this is an array of length m of the relative weights of the corresponding wavelengths in incident_wavelength.

In the case of a polychromatic beam that varies shot-to-shot, this is a 2D array of dimensions nP by m (slow to fast) of the relative weights of the corresponding wavelengths in incident_wavelength.

incident_wavelength_weights: (optional) NX_FLOAT

In the case of a polychromatic beam this is an array of length m of the relative weights of the corresponding wavelengths in incident_wavelength.
In the case of a polychromatic beam that varies shot-to-shot, this is a 2D array of dimensions \( n_p \) by \( m \) (slow to fast) of the relative weights of the corresponding wavelengths in \( \text{incident}_\text{wavelength} \).

\textbf{incident}_\text{wavelength}_\text{spread}: (optional) \( \text{NX_FLOAT} \) 
\{\text{units=} \text{NX\_WAVELENGTH}\}

The wavelength spread FWHM for the corresponding wavelength(s) in \( \text{incident}_\text{wavelength} \).

In the case of shot-to-shot variation in the wavelength spread, this is a 2D array of dimension \( n_P \) by \( m \) (slow to fast) of the spreads of the corresponding wavelengths in \( \text{incident}_\text{wavelength} \).

\textbf{flux}: (optional) \( \text{NX\_FLOAT} \) \{\text{units=} \text{NX\_FLUX}\}

Flux density incident on beam plane area in photons per second per unit area.

In the case of a beam that varies in flux shot-to-shot, this is an array of values, one for each recorded shot.

\textbf{total}\_\text{flux}: (optional) \( \text{NX\_FLOAT} \) \{\text{units=} \text{NX\_FREQUENCY}\}

Flux incident on beam plane in photons per second. In other words this is the \( \text{flux} \) integrated over area.

Useful where spatial beam profiles are not known.

In the case of a beam that varies in total flux shot-to-shot, this is an array of values, one for each recorded shot.

\textbf{flux}_\text{integrated}: (optional) \( \text{NX\_FLOAT} \) \{\text{units=} \text{NX\_PER\_AREA}\}

Flux density incident on beam plane area in photons per unit area. In other words this is the \( \text{flux} \) integrated over time.

Useful where temporal beam profiles of flux are not known.

In the case of a beam that varies in flux shot-to-shot, this is an array of values, one for each recorded shot.

\textbf{total}\_\text{flux}_\text{integrated}: (optional) \( \text{NX\_FLOAT} \) \{\text{units=} \text{NX\_DIMENSIONLESS}\}

Flux incident on beam plane in photons. In other words this is the \( \text{flux} \) integrated over time and area.

Useful where temporal beam profiles of flux are not known.

In the case of a beam that varies in total flux shot-to-shot, this is an array of values, one for each recorded shot.

\textbf{incident}\_\text{beam}\_\text{size}: (recommended) \( \text{NX\_FLOAT} \) (Rank: 1, Dimensions: [2]) \{\text{units=} \text{NX\_LENGTH}\}

Two-element array of FWHM (if Gaussian or Airy function) or diameters (if top hat) or widths (if rectangular) of the beam in the order \( x \), \( y \)

\textbf{profile}: (recommended) \( \text{NX\_CHAR} \)

The beam profile, Gaussian, Airy function, top-hat or rectangular. The profile is given in the plane of incidence of the beam on the sample.
Any of these values: Gaussian | Airy | top-hat | rectangular

**incident_polarisation_stokes**: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [nP, 4])

**DEPRECATED**: use incident_polarization_stokes, see [https://github.com/nexusformat/definitions/issues/708](https://github.com/nexusformat/definitions/issues/708)

Polarization vector on entering beamline component using Stokes notation

**incident_polarization_stokes**: (recommended) *NX_NUMBER* (Rank: 2, Dimensions: [nP, 4])

Polarization vector on entering beamline component using Stokes notation. See incident_polarization_stokes in *NXbeam*

**incident_wavelength_spectrum**: (optional) *NXdata*

**SOURCE**: (required) *NXsource*

The neutron or x-ray storage ring/facility. Note, the NXsource base class has many more fields available, but at present we only require the name.

**name**: (required) *NX_CHAR*

Name of source. Consistency with the naming in [https://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v50.dic/Items/_diffrn_source.pdbx_synchrotron_site.html](https://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v50.dic/Items/_diffrn_source.pdbx_synchrotron_site.html) controlled vocabulary is highly recommended.

**@short_name**: (optional) *NX_CHAR*

short name for source, perhaps the acronym

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXmx/ENTRY-group
- /NXmx/ENTRY/DATA-group
- /NXmx/ENTRY/DATA/data-field
- /NXmx/ENTRY/definition-field
- /NXmx/ENTRY/end_time-field
- /NXmx/ENTRY/end_time_estimated-field
- /NXmx/ENTRY/INSTRUMENT-group
- /NXmx/ENTRY/INSTRUMENT/ATTENUATOR-group
- /NXmx/ENTRY/INSTRUMENT/ATTENUATOR/attenuator_transmission-field
- /NXmx/ENTRY/INSTRUMENT/BEAM-group
- /NXmx/ENTRY/INSTRUMENT/BEAM/flux-field
- /NXmx/ENTRY/INSTRUMENT/BEAM/flux_integrated-field
- /NXmx/ENTRY/INSTRUMENT/BEAM/incident_beam_size-field
- /NXmx/ENTRY/INSTRUMENT/BEAM/incident_polarisation_stokes-field
• /NXmx/ENTRY/INSTRUMENT/BEAM/incident_polarization_stokes-field
• /NXmx/ENTRY/INSTRUMENT/BEAM/incident_wavelength-field
• /NXmx/ENTRY/INSTRUMENT/BEAM/incident_wavelength_spectrum-group
• /NXmx/ENTRY/INSTRUMENT/BEAM/incident_wavelength_spread-field
• /NXmx/ENTRY/INSTRUMENT/BEAM/incident_wavelength_weight-field
• /NXmx/ENTRY/INSTRUMENT/BEAM/incident_wavelength_weights-field
• /NXmx/ENTRY/INSTRUMENT/BEAM/profile-field
• /NXmx/ENTRY/INSTRUMENT/BEAM/total_flux-field
• /NXmx/ENTRY/INSTRUMENT/BEAM/total_flux_integrated-field
• /NXmx/ENTRY/INSTRUMENT/BEAM/@flux-attribute
• /NXmx/ENTRY/INSTRUMENT/DETECTOR-group
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/angular_calibration-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/angular_calibration_applied-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/beam_center_derived-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/beam_center_x-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/beam_center_y-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/bit_depth_readout-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/COLLECTION-group
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/count_time-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/countrate_correction_applied-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/data-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/dead_time-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/depends_on-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/description-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE-group
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/data_origin-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/data_size-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/data_stride-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/fast_pixel_direction-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/fast_pixel_direction@depends_on-attribute
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/fast_pixel_direction@offset-attribute
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/fast_pixel_direction@transformation_type-attribute
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/fast_pixel_direction@vector-attribute
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/module_offset-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/module_offset@depends_on-attribute

3.3. NeXus Class Definitions 373
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/module_offset@offset-attribute
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/module_offset@transformation_type-attribute
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/slow_pixel_direction-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/slow_pixel_direction@depends_on-attribute
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/slow_pixel_direction@offset-attribute
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/slow_pixel_direction@transformation_type-attribute
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/DETECTOR_MODULE/slow_pixel_direction@vector-attribute
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/detector_readout_time-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/distance-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/distance_derived-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/flatfield-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/flatfield_applied-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/flatfield_error-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/flatfield_errors-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/frame_time-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/gain_setting-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/pixel_mask-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/pixel_mask_applied-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/saturation_value-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/sensor_material-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/sensor_thickness-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/threshold_energy-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/time_per_channel-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/TRANSFORMATIONS-group
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/type-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/underload_value-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR/virtual_pixel_interpolation_applied-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR_GROUP-group
• /NXmx/ENTRY/INSTRUMENT/DETECTOR_GROUP/group_index-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR_GROUP/group_names-field
• /NXmx/ENTRY/INSTRUMENT/DETECTOR_GROUP/group_parent-field
• /NXmx/ENTRY/INSTRUMENT/name-field
• /NXmx/ENTRY/INSTRUMENT/name@short_name-attribute
NXrefscan

Status:
application definition, extends NXobject

Description:
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.

Symbols:
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:
NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXmonochromator, NXsample, NXsource

Structure:

entry: (required) NXentry
    title: (required) NX_CHAR
    start_time: (required) NX_DATE_TIME
    end_time: (required) NX_DATE_TIME
    definition: (required) NX_CHAR
        Official NeXus NDXL schema to which this file conforms
        Obligatory value: NXrefscan
    instrument: (required) NXinstrument
    SOURCE: (required) NXsource

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXmx.nxdl.xml
**type**: (required) *NX_CHAR*

**name**: (required) *NX_CHAR*

**probe**: (required) *NX_CHAR*

Any of these values: neutron|x-ray|electron

**monochromator**: (required) *NXmonochromator*

**wavelength**: (required) *NX_FLOAT* \{units=*NX_WAVELENGTH*\}

**DETECTOR**: (required) *NXdetector*

**data**: (required) *NX_INT* (Rank: 1, Dimensions: [nP])

**polar_angle**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) \{units=*NX_ANGLE*\}

**sample**: (required) *NXSample*

**name**: (required) *NX_CHAR*

Descriptive name of sample

**rotation_angle**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) \{units=*NX_ANGLE*\}

**control**: (required) *NXmonitor*

**mode**: (required) *NX_CHAR*

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: monitor|timer

**preset**: (required) *NX_FLOAT*

preset value for time or monitor

**data**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) \{units=*NX_ANY*\}

Monitor counts for each step

**data**: (required) *NXdata*

**data**: *link* (suggested target: /NXentry/NXinstrument/NXdetector/data)

**rotation_angle**: *link* (suggested target: /NXentry/NXsample/rotation_angle)

**polar_angle**: *link* (suggested target: /NXentry/NXinstrument/NXdetector/polar_angle)

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXrefscan/entry-group
- /NXrefscan/entry/control-group
- /NXrefscan/entry/control/data-field
- /NXrefscan/entry/control/mode-field
- /NXrefscan/entry/control/preset-field
• /NXrefscan/entry/data-group
• /NXrefscan/entry/data/data-link
• /NXrefscan/entry/data/polar_angle-link
• /NXrefscan/entry/data/rotation_angle-link
• /NXrefscan/entry/definition-field
• /NXrefscan/entry/end_time-field
• /NXrefscan/entry/instrument-group
• /NXrefscan/entry/instrument/DETECTOR-group
• /NXrefscan/entry/instrument/DETECTOR/data-field
• /NXrefscan/entry/instrument/DETECTOR/polar_angle-field
• /NXrefscan/entry/instrument/monochromator-group
• /NXrefscan/entry/instrument/monochromator/wavelength-field
• /NXrefscan/entry/instrument/SOURCE-group
• /NXrefscan/entry/instrument/SOURCE/name-field
• /NXrefscan/entry/instrument/SOURCE/probe-field
• /NXrefscan/entry/instrument/SOURCE/type-field
• /NXrefscan/entry/sample-group
• /NXrefscan/entry/sample/name-field
• /NXrefscan/entry/sample/rotation_angle-field
• /NXrefscan/entry/start_time-field
• /NXrefscan/entry/title-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXrefscan.nxdl.xml

NXrefscanto

Status:
application definition, extends NXobject

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

Symbols:
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

xSize: xSize description
ySize: ySize description
nTOF: nTOF description

Groups cited:
NXdata, NXdetector, NXdisk_chopper, NXentry, NXinstrument, NXmonitor, NXsample

Structure:
entry: (required) **NXentry**

  title: (required) **NX_CHAR**

  start_time: (required) **NX_DATE_TIME**

  end_time: (required) **NX_DATE_TIME**

  definition: (required) **NX_CHAR**

    Official NeXus NXDL schema to which this file conforms
    Obligatory value: **NXreftof**

instrument: (required) **NXinstrument**

  name: (required) **NX_CHAR**

chopper: (required) **NXdisk_chopper**

  distance: (required) **NX_FLOAT** {units=**NX_LENGTH**}

    Distance between chopper and sample

detector: (required) **NXdetector**

  data: (required) **NX_INT** (Rank: 3, Dimensions: [xSize, ySize, nTOF])

  time_of_flight: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nTOF])
    {units=**NX_TIME_OF_FLIGHT**}

    Array of time values for each bin in a time-of-flight measurement

  distance: (required) **NX_FLOAT** {units=**NX_LENGTH**}

  polar_angle: (required) **NX_FLOAT** {units=**NX_ANGLE**}

  x_pixel_size: (required) **NX_FLOAT** {units=**NX_LENGTH**}

  y_pixel_size: (required) **NX_FLOAT** {units=**NX_LENGTH**}

sample: (required) **NXsample**

  name: (required) **NX_CHAR**

    Descriptive name of sample

  rotation_angle: (required) **NX_FLOAT** {units=**NX_ANGLE**}

control: (required) **NXmonitor**

  mode: (required) **NX_CHAR**

    Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

    Any of these values: monitor | timer

  preset: (required) **NX_FLOAT** {units=**NX_ANY**}

    preset value for time or monitor

  integral: (required) **NX_INT**

    Total integral monitor counts

  time_of_flight: (required) **NX_FLOAT** {units=**NX_TIME_OF_FLIGHT**}

    Time channels

  data: (required) **NX_INT**
Monitor counts in each time channel

**data:** (required) `NXdata`

**data:** `link` (suggested target: `/NXentry/NXinstrument/NXdetector/data`)

**time_of_flight:** `link` (suggested target: `/NXentry/NXinstrument/NXdetector/time_of_flight`)

### Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXreftof/entry-group`
- `/NXreftof/entry/control-group`
- `/NXreftof/entry/control/data-field`
- `/NXreftof/entry/control/integral-field`
- `/NXreftof/entry/control/mode-field`
- `/NXreftof/entry/control/preset-field`
- `/NXreftof/entry/control/time_of_flight-field`
- `/NXreftof/entry/data-group`
- `/NXreftof/entry/data/data-link`
- `/NXreftof/entry/data/time_of_flight-link`
- `/NXreftof/entry/definition-field`
- `/NXreftof/entry/end_time-field`
- `/NXreftof/entry/instrument-group`
- `/NXreftof/entry/instrument/chopper-group`
- `/NXreftof/entry/instrument/chopper/distance-field`
- `/NXreftof/entry/instrument/detector-group`
- `/NXreftof/entry/instrument/detector/data-field`
- `/NXreftof/entry/instrument/detector/distance-field`
- `/NXreftof/entry/instrument/detector/polar_angle-field`
- `/NXreftof/entry/instrument/detector/time_of_flight-field`
- `/NXreftof/entry/instrument/detector/x_pixel_size-field`
- `/NXreftof/entry/instrument/detector/y_pixel_size-field`
- `/NXreftof/entry/instrument/name-field`
- `/NXreftof/entry/sample-group`
- `/NXreftof/entry/sample/name-field`
- `/NXreftof/entry/sample/rotation_angle-field`
- `/NXreftof/entry/start_time-field`
- `/NXreftof/entry/title-field`
NXsas

Status:

application definition, extends NXobject

Description:

Raw, monochromatic 2-D SAS data with an area detector.

This is an application definition for raw data (not processed or reduced data) from a 2-D small angle scattering instrument collected with a monochromatic beam and an area detector. It is meant to be suitable both for neutron SANS and X-ray SAXS data.

It covers all raw data from any monochromatic SAS techniques that use an area detector: SAS, WSAS, grazing incidence, GISAS

It covers all raw data from any SAS techniques that use an area detector and a monochromatic beam.

Symbols:

The symbol(s) listed here will be used below to coordinate fields with the same shape.

nXPixel: Number of pixels in x direction.
nYPixel: Number of pixels in y direction.

Groups cited:

NXcollimator, NXdata, NXdetector, NXentry, NXgeometry, NXinstrument, NXmonitor, NXmonochromator, NXsample, NXshape, NXsource

Structure:

ENTRY: (required) NXentry

   title: (optional) NX_CHAR
   start_time: (optional) NX_DATE_TIME
   end_time: (optional) NX_DATE_TIME
   definition: (required) NX_CHAR

       Official NeXus NXDL schema to which this file conforms.

       Obligatory value: NXsas

INSTRUMENT: (required) NXinstrument

   name: (required) NX_CHAR

       Name of the instrument actually used to perform the experiment.

SOURCE: (required) NXsource

   type: (required) NX_CHAR

       Type of radiation source.

   name: (optional) NX_CHAR

       Name of the radiation source.

   probe: (required) NX_CHAR
Name of radiation probe, necessary to compute the sample contrast.

Any of these values: neutron | x-ray

**MONOCHROMATOR**: (required) *NXmonochromator*

**wavelength**: (required) *NX_FLOAT {units=NX_WAVELENGTH}*

The wavelength ($\lambda$) of the radiation.

**wavelength_spread**: (optional) *NX_FLOAT*

delta_lambda/lambda ($\Delta \lambda/\lambda$): Important for resolution calculations.

**COLLIMATOR**: (optional) *NXcollimator*

**GEOMETRY**: (required) *NXgeometry*

**SHAPE**: (required) *NXshape*

**shape**: (required) *NX_CHAR*

Any of these values: nxcylinder | nxbox

**size**: (required) *NX_FLOAT {units=NX_LENGTH}*

The collimation length.

**DETECTOR**: (required) *NXdetector*

**data**: (required) *NX_NUMBER (Rank: 2, Dimensions: [nXPixel, nYPixel]*)

This is area detector data, 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.

It is necessary to define which are the x and y directions, to coordinate with the pixel size and compute Q.

**distance**: (required) *NX_FLOAT {units=NX_LENGTH}*

The distance between detector and sample.

**x_pixel_size**: (required) *NX_FLOAT {units=NX_LENGTH}*

Physical size of a pixel in x-direction.

**y_pixel_size**: (required) *NX_FLOAT {units=NX_LENGTH}*

Physical size of a pixel in y-direction.

**polar_angle**: (optional) *NX_FLOAT {units=NX_ANGLE}*

**azimuthal_angle**: (optional) *NX_FLOAT {units=NX_ANGLE}*

**rotation_angle**: (optional) *NX_FLOAT {units=NX_ANGLE}*

**aequatorial_angle**: (optional) *NX_FLOAT {units=NX_ANGLE}*

**beam_center_x**: (optional) *NX_FLOAT {units=NX_LENGTH}*

This is the x position where the direct beam would hit the detector. This is a length, not a pixel position, and can be outside of the actual detector.

It is expected that data reduction will determine beam center from the raw data, thus it is not required here. The instrument can provide an initial or nominal value to advise data reduction.
**beam_center_y**: (optional) \texttt{NX_FLOAT} \{\texttt{units=NX\_LENGTH}\}

This is the y position where the direct beam would hit the detector. This is a length, not a pixel position, and can be outside of the actual detector.

It is expected that data reduction will determine beam center from the raw data, thus it is not required here. The instrument can provide an initial or nominal value to advise data reduction.

**SAMPLE**: (optional) \texttt{NXsample}

\textbf{name}: (required) \texttt{NX\_CHAR}

Descriptive name of sample.

\textbf{aequatorial_angle}: (optional) \texttt{NX_FLOAT} \{\texttt{units=NX\_ANGLE}\}

**MONITOR**: (optional) \texttt{NXmonitor}

\textbf{mode}: (required) \texttt{NX\_CHAR}

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: \texttt{monitor} | \texttt{timer}

\textbf{preset}: (required) \texttt{NX\_FLOAT}

Preset value for time or monitor.

\textbf{integral}: (required) \texttt{NX\_FLOAT} \{\texttt{units=NX\_ANY}\}

Total integral monitor counts.

**DATA**: (required) \texttt{NXdata}

\textbf{@signal}: (optional) \texttt{NX\_CHAR}

Name the \textit{plottable} field. The link here defines this name as \texttt{data}.

Obligatory value: \texttt{data}

\textbf{data}: \textit{link} (suggested target: /NXentry/NXinstrument/NXdetector/data)

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXsas/ENTRY-group
- /NXsas/ENTRY/DATA-group
- /NXsas/ENTRY/DATA/data-link
- /NXsas/ENTRY/DATA/@signal-attribute
- /NXsas/ENTRY/definition-field
- /NXsas/ENTRY/end_time-field
- /NXsas/ENTRY/INSTRUMENT-group
- /NXsas/ENTRY/INSTRUMENT/COLLIMATOR-group
- /NXsas/ENTRY/INSTRUMENT/COLLIMATOR/GEOMETRY-group
- /NXsas/ENTRY/INSTRUMENT/COLLIMATOR/GEOMETRY/SHAPE-group
• /NXsas/ENTRY/INSTRUMENT/COLLMATOR/GEOMETRY/SHAPE/shape-field
• /NXsas/ENTRY/INSTRUMENT/COLLMATOR/GEOMETRY/SHAPE/size-field
• /NXsas/ENTRY/INSTRUMENT/DETECTOR-group
• /NXsas/ENTRY/INSTRUMENT/DETECTOR/aequatorial_angle-field
• /NXsas/ENTRY/INSTRUMENT/DETECTOR/azimuthal_angle-field
• /NXsas/ENTRY/INSTRUMENT/DETECTOR/beam_center_x-field
• /NXsas/ENTRY/INSTRUMENT/DETECTOR/beam_center_y-field
• /NXsas/ENTRY/INSTRUMENT/DETECTOR/data-field
• /NXsas/ENTRY/INSTRUMENT/DETECTOR/distance-field
• /NXsas/ENTRY/INSTRUMENT/DETECTOR/polar_angle-field
• /NXsas/ENTRY/INSTRUMENT/DETECTOR/rotation_angle-field
• /NXsas/ENTRY/INSTRUMENT/DETECTOR/x_pixel_size-field
• /NXsas/ENTRY/INSTRUMENT/DETECTOR/y_pixel_size-field
• /NXsas/ENTRY/INSTRUMENT/MONOCHROMATOR-group
• /NXsas/ENTRY/INSTRUMENT/MONOCHROMATOR/wavelength-field
• /NXsas/ENTRY/INSTRUMENT/MONOCHROMATOR/wavelength_spread-field
• /NXsas/ENTRY/INSTRUMENT/name-field
• /NXsas/ENTRY/INSTRUMENT/SOURCE-group
• /NXsas/ENTRY/INSTRUMENT/SOURCE/name-field
• /NXsas/ENTRY/INSTRUMENT/SOURCE/probe-field
• /NXsas/ENTRY/INSTRUMENT/SOURCE/type-field
• /NXsas/ENTRY/MONITOR-group
• /NXsas/ENTRY/MONITOR/integral-field
• /NXsas/ENTRY/MONITOR/mode-field
• /NXsas/ENTRY/MONITOR/preset-field
• /NXsas/ENTRY/SAMPLE-group
• /NXsas/ENTRY/SAMPLE/aequatorial_angle-field
• /NXsas/ENTRY/SAMPLE/name-field
• /NXsas/ENTRY/SAMPLE/start_time-field
• /NXsas/ENTRY/title-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXsas.nxdl.xml
**NXsastof**

**Status:**
application definition, extends *NXobject*

**Description:**
raw, 2-D SAS data with an area detector with a time-of-flight source
It covers all raw data from any SAS techniques that use an area detector at a time-of-flight source.

**Symbols:**
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

- **nXPix**: nXPix description
- **nYPix**: nYPix description
- **nTOF**: nTOF description

**Groups cited:**
NXcollimator, NXdata, NXdetector, NXentry, NXgeometry, NXinstrument, NXmonitor, NXsample, NXshape, NXsource

**Structure:**
ENTRY: (required) NXentry

- **@entry**: (required) *NX_CHAR*
  NeXus convention is to use “entry1”, “entry2”, … for analysis software to locate each entry
- **title**: (required) *NX_CHAR*
- **start_time**: (required) *NX_DATE_TIME*
- **definition**: (required) *NX_CHAR*
  Official NeXus NXDL schema to which this file conforms
  Obligatory value: NXsastof
- **instrument**: (required) NXinstrument
  - **name**: (required) *NX_CHAR*
    Name of the instrument actually used to perform the experiment
- **source**: (required) NXsource
  - **type**: (required) *NX_CHAR*
    type of radiation source
  - **name**: (required) *NX_CHAR*
    Name of the radiation source
  - **probe**: (required) *NX_CHAR*
    Any of these values: neutron | x-ray
- **collimator**: (required) NXcollimator
- **geometry**: (required) NXgeometry
- **shape**: (required) NXshape
shape: (required) **NX_CHAR**

Any of these values: nxcylinder | nxbox

size: (required) **NX_FLOAT** {units=**NX_LENGTH**}

The collimation length
detector: (required) **NXdetector**
data: (required) **NX_NUMBER** (Rank: 3, Dimensions: [nXPixel, nYPixel, nTOF])

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.
time_of_flight: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nTOF]) {units=**NX_TIME_OF_FLIGHT**}
distance: (required) **NX_FLOAT** {units=**NX_LENGTH**}
The distance between detector and sample
x_pixel_size: (required) **NX_FLOAT** {units=**NX_LENGTH**}
Physical size of a pixel in x-direction
y_pixel_size: (required) **NX_FLOAT** {units=**NX_LENGTH**}
Size of a pixel in y direction
polar_angle: (required) **NX_FLOAT** {units=**NX_ANGLE**}
azimuthal_angle: (required) **NX_FLOAT** {units=**NX_ANGLE**}
rotation_angle: (required) **NX_FLOAT** {units=**NX_ANGLE**}
aequatorial_angle: (required) **NX_FLOAT** {units=**NX_ANGLE**}
beam_center_x: (required) **NX_FLOAT** {units=**NX_LENGTH**}
This is the x position where the direct beam would hit the detector. This is a length, not a pixel position, and can be outside of the actual detector.
beam_center_y: (required) **NX_FLOAT** {units=**NX_LENGTH**}
This is the y position where the direct beam would hit the detector. This is a length, not a pixel position, and can be outside of the actual detector.
sample: (required) **NXsample**
name: (required) **NX_CHAR**
Descriptive name of sample
eaquatorial_angle: (required) **NX_FLOAT** {units=**NX_ANGLE**}
control: (required) **NXmonitor**
mode: (required) **NX_CHAR**
Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: monitor | timer
preset: (required) **NX_FLOAT**
preset value for time or monitor

**data:** (required) *NX_INT* (Rank: 1, Dimensions: [nTOF])

**time_of_flight:** (required) *NX_FLOAT* (Rank: 1, Dimensions: [nTOF])

{units=*NX_TIME_OF_FLIGHT*}

**data:** (required) *NXdata*

**data:** *link* (suggested target: /NXentry/NXinstrument/NXdetector/data)

**time_of_flight:** *link* (suggested target: /NXentry/NXinstrument/NXdetector/time_of_flight)

### Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXsastof/ENTRY-group
- /NXsastof/ENTRY/control-group
- /NXsastof/ENTRY/control/data-field
- /NXsastof/ENTRY/control/mode-field
- /NXsastof/ENTRY/control/preset-field
- /NXsastof/ENTRY/control/time_of_flight-field
- /NXsastof/ENTRY/data-group
- /NXsastof/ENTRY/data/data-link
- /NXsastof/ENTRY/data/time_of_flight-link
- /NXsastof/ENTRY/definition-field
- /NXsastof/ENTRY/instrument-group
- /NXsastof/ENTRY/instrument/collimator-group
- /NXsastof/ENTRY/instrument/collimator/geometry-group
- /NXsastof/ENTRY/instrument/collimator/geometry/shape-group
- /NXsastof/ENTRY/instrument/collimator/geometry/shape/shape-field
- /NXsastof/ENTRY/instrument/collimator/geometry/shape/size-field
- /NXsastof/ENTRY/instrument/detector-group
- /NXsastof/ENTRY/instrument/detector/aequatorial_angle-field
- /NXsastof/ENTRY/instrument/detector/azimuthal_angle-field
- /NXsastof/ENTRY/instrument/detector/beam_center_x-field
- /NXsastof/ENTRY/instrument/detector/beam_center_y-field
- /NXsastof/ENTRY/instrument/detector/data-field
- /NXsastof/ENTRY/instrument/detector/distance-field
- /NXsastof/ENTRY/instrument/detector/polar_angle-field
- /NXsastof/ENTRY/instrument/detector/rotation_angle-field
NXscan

Status:
application definition, extends NXobject

Description:
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, the symbol $NP$ is used to represent 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 exist 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.

Symbols:
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

xDim: xDim description

yDim: yDim description

Groups cited:

NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample

Structure:

ENTRY: (required) NXentry

  title: (required) NX_CHAR

  start_time: (required) NX_DATE_TIME

  end_time: (required) NX_DATE_TIME

  definition: (required) NX_CHAR

    Official NeXus NXDL schema to which this file conforms

    Obligatory value: NXscan

INSTRUMENT: (required) NXinstrument

DETECTOR: (required) NXdetector

  data: (required) NX_INT (Rank: 3, Dimensions: [nP, xDim, yDim])

SAMPLE: (required) NXsample

  rotation_angle: (required) NX_FLOAT (Rank: 1, Dimensions: [nP])

MONITOR: (required) NXmonitor

  data: (required) NX_INT (Rank: 1, Dimensions: [nP])

DATA: (required) NXdata

  data: link (suggested target: /NXentry/NXinstrument/NXdetector/data)

  rotation_angle: link (suggested target: /NXentry/NXsample/rotation_angle)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXscan/ENTRY-group
- /NXscan/ENTRY/DATA-group
- /NXscan/ENTRY/DATA/data-link
- /NXscan/ENTRY/DATA/rotation_angle-link
- /NXscan/ENTRY/definition-field
- /NXscan/ENTRY/end_time-field
- /NXscan/ENTRY/INSTRUMENT-group
- /NXscan/ENTRY/INSTRUMENT/DETECTOR-group
- /NXscan/ENTRY/INSTRUMENT/DETECTOR/data-field
NXscan

• /NXscan/ENTRY/MONITOR-group
• /NXscan/ENTRY/MONITOR/data-field
• /NXscan/ENTRY/SAMPLE-group
• /NXscan/ENTRY/SAMPLE/rotation_angle-field
• /NXscan/ENTRY/start_time-field
• /NXscan/ENTRY/title-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXscan.nxdl.xml

NXspe

Status:
application definition, extends NXobject

Description:
NXSPE Inelastic Format. Application definition for NXSPE file format.

Symbols:
No symbol table

Groups cited:
NXcollection, NXdata, NXentry, NXfermi_chopper, NXinstrument, NXsample

Structure:
ENTRY: (required) NXentry

  program_name: (required) NX_CHAR

  definition: (required) NX_CHAR

    Official NeXus NXDL schema to which this file conforms.

    Obligatory value: NXSPE

    @version: (required) NX_CHAR

  NXSPE_info: (required) NXcollection

    fixed_energy: (required) NX_FLOAT {units=NX_ENERGY}

      The fixed energy used for this file.

    ki_over_kf_scaling: (required) NX_BOOLEAN

      Indicates whether ki/kf scaling has been applied or not.

    psi: (required) NX_FLOAT {units=NX_ANGLE}

      Orientation angle as expected in DCS-MSlice

  data: (required) NXdata

    azimuthal: (required) NX_FLOAT {units=NX_ANGLE}

    azimuthal_width: (required) NX_FLOAT {units=NX_ANGLE}

    polar: (required) NX_FLOAT {units=NX_ANGLE}

    polar_width: (required) NX_FLOAT {units=NX_ANGLE}
distance: (required) NX_FLOAT {units=NX_LENGTH}
data: (required) NX_NUMBER
error: (required) NX_NUMBER
energy: (required) NX_FLOAT {units=NX_ENERGY}

INSTRUMENT: (required) NXinstrument
  name: (required) NX_CHAR
  FERMI_CHOPPER: (required) NXfermi_chopper
    energy: (required) NX_NUMBER {units=NX_ENERGY}

SAMPLE: (required) NXsample
  rotation_angle: (required) NX_NUMBER {units=NX_ANGLE}
  seblock: (required) NX_CHAR
  temperature: (required) NX_NUMBER {units=NX_TEMPERATURE}

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXspe/ENTRY-group
- /NXspe/ENTRY/data-group
- /NXspe/ENTRY/data/azimuthal-field
- /NXspe/ENTRY/data/azimuthal_width-field
- /NXspe/ENTRY/data/data-field
- /NXspe/ENTRY/data/distance-field
- /NXspe/ENTRY/data/energy-field
- /NXspe/ENTRY/data/error-field
- /NXspe/ENTRY/data/polar-field
- /NXspe/ENTRY/data/polar_width-field
- /NXspe/ENTRY/definition-field
- /NXspe/ENTRY/definition@version-attribute
- /NXspe/ENTRY/INSTRUMENT-group
- /NXspe/ENTRY/INSTRUMENT/FERMI_CHOPPER-group
- /NXspe/ENTRY/INSTRUMENT/FERMI_CHOPPER/energy-field
- /NXspe/ENTRY/INSTRUMENT/name-field
- /NXspe/ENTRY/NXSPE_info-group
- /NXspe/ENTRY/NXSPE_info/fixed_energy-field
- /NXspe/ENTRY/NXSPE_info/ki_over_kf_scaling-field
- /NXspe/ENTRY/NXSPE_info/psi-field
- /NXspe/ENTRY/program_name-field
NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXspe.nxdl.xml

NXsqom

Status:
application definition, extends NXobject

Description:
This is the application definition for S(Q,OM) processed data.
As this kind of data is in general not on a rectangular grid after data reduction, it is stored as Q,E positions
plus their intensity, table like. It is the task of a possible visualisation program to regrid this data in a
sensible way.

Symbols:
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:
NXdata, NXentry, NXinstrument, NXparameters, NXprocess, NXsample, NXsource

Structure:

ENTRY: (required) NXentry
  @entry: (required) NX_CHAR
  title: (required) NX_CHAR
  definition: (required) NX_CHAR
    Official NeXus NXDL schema to which this file conforms
    Obligatory value: NXsqom

instrument: (required) NXinstrument
  name: (required) NX_CHAR
    Name of the instrument from which this data was reduced.

SOURCE: (required) NXsource
  type: (required) NX_CHAR
  name: (required) NX_CHAR
  probe: (required) NX_CHAR
    Any of these values: neutron | x-ray | electron

SAMPLE: (required) NXsample
  name: (required) NX_CHAR
Descriptive name of sample

**reduction**: (required) *NXprocess*

**program**: (required) *NX_CHAR*

**version**: (required) *NX_CHAR*

**input**: (required) *NXparameters*

Input parameters for the reduction program used

**filenames**: (required) *NX_CHAR*

Raw data files used to generate this I(Q)

**output**: (required) *NXparameters*

Eventual output parameters from the data reduction program used

**DATA**: (required) *NXdata*

**data**: (required) *NX_INT* (Rank: 1, Dimensions: [nP])

This is the intensity for each point in QE

**qx**: (required) *NX_NUMBER* (Rank: 1, Dimensions: [nP]) {units=*NX_WAVENUMBER*}

Positions for the first dimension of Q

**qy**: (required) *NX_NUMBER* (Rank: 1, Dimensions: [nP]) {units=*NX_WAVENUMBER*}

Positions for the the second dimension of Q

**qz**: (required) *NX_NUMBER* (Rank: 1, Dimensions: [nP]) {units=*NX_WAVENUMBER*}

Positions for the the third dimension of Q

**en**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ENERGY*}

Values for the energy transfer for each point

---

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXsqom/ENTRY-group
- /NXsqom/ENTRY/DATA-group
- /NXsqom/ENTRY/DATA/data-field
- /NXsqom/ENTRY/DATA/en-field
- /NXsqom/ENTRY/DATA/qx-field
- /NXsqom/ENTRY/DATA/qy-field
- /NXsqom/ENTRY/DATA/qz-field
- /NXsqom/ENTRY/definition-field
- /NXsqom/ENTRY/instrument-group
- /NXsqom/ENTRY/instrument/name-field
• /NXsqom/ENTRY/instrument/SOURCE-group
• /NXsqom/ENTRY/instrument/SOURCE/name-field
• /NXsqom/ENTRY/instrument/SOURCE/probe-field
• /NXsqom/ENTRY/instrument/SOURCE/type-field
• /NXsqom/ENTRY/reduction-group
• /NXsqom/ENTRY/reduction/input-group
• /NXsqom/ENTRY/reduction/input/filenames-field
• /NXsqom/ENTRY/reduction/output-group
• /NXsqom/ENTRY/reduction/program-field
• /NXsqom/ENTRY/reduction/version-field
• /NXsqom/ENTRY/SAMPLE-group
• /NXsqom/ENTRY/SAMPLE/name-field
• /NXsqom/ENTRY/title-field
• /NXsqom/ENTRY@entry-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXsqom.nxdl.xml

NXstxm

Status:
application definition, extends NXobject

Description:
Application definition for a STXM instrument.

The interferometer position measurements, monochromator photon energy values and detector measurements are all treated as NXdetectors and stored within the NXinstrument group as lists of values stored in chronological order. The NXdata group then holds another version of the data in a regular 3D array (NumE by NumY by NumX, for a total of nP points in a sample image stack type scan). The former data values should be stored with a minimum loss of precision, while the latter values can be simplified and/or approximated in order to fit the constraints of a regular 3D array. ‘Line scans’ and ‘point spectra’ are just sample_image scan types with reduced dimensions in the same way as single images have reduced E dimensions compared to image ‘stacks’.

Symbols:
These symbols will be used below to coordinate the shapes of the datasets.

nP: Total number of scan points
nE: Number of photon energies scanned
nX: Number of pixels in X direction
nY: Number of pixels in Y direction
detectorRank: Rank of data array provided by the detector for a single measurement

Groups cited:
NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXmonochromator, NXsample, NXsource
Structure:

**ENTRY**: (required) *NXentry*

  - **title**: (required) *NX_CHAR*
  - **start_time**: (required) *NX_DATE_TIME*
  - **end_time**: (required) *NX_DATE_TIME*
  - **definition**: (required) *NX_CHAR*
    
      Official NeXus NXDL schema to which this file conforms
      
      Obligatory value: *NXstxm*

**INSTRUMENT**: (required) *NXinstrument*

**SOURCE**: (required) *NXsource*

  - **type**: (required) *NX_CHAR*
  - **name**: (required) *NX_CHAR*
  - **probe**: (required) *NX_CHAR*

**monochromator**: (required) *NXmonochromator*

  - **energy**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP])

**DETECTOR**: (required) *NXdetector*

  - **data**: (required) *NX_NUMBER* (Rank: 1+detectorRank, Dimensions: [nP])

  **sample_x**: (optional) *NXdetector*

    Measurements of the sample position from the x-axis interferometer.
    
    **data**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP])

  **sample_y**: (optional) *NXdetector*

    Measurements of the sample position from the y-axis interferometer.
    
    **data**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP])

  **sample_z**: (optional) *NXdetector*

    Measurements of the sample position from the z-axis interferometer.
    
    **data**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP])

**SAMPLE**: (required) *NXsample*

  - **rotation_angle**: (required) *NX_FLOAT*

**DATA**: (required) *NXdata*

  - **stxm_scan_type**: (required) *NX_CHAR*

    Label for typical scan types as a convenience for humans.
    
    Each label corresponds to a specific set of axes being scanned to produce a data array of shape:
    
    - sample point spectrum: (photon_energy,)
    - sample line spectrum: (photon_energy, sample_y/sample_x)
    - sample image: (sample_y, sample_x)
    - sample image stack: (photon_energy, sample_y, sample_x)
• sample focus: (zoneplate.z, sample.y/sample.x)
• osa image: (osa.y, osa.x)
• osa focus: (zoneplate.z, osa.y/osa.x)
• detector image: (detector.y, detector.x)

The “generic scan” string is to be used when none of the other choices are appropriate.

Any of these values:
• sample point spectrum
• sample line spectrum
• sample image
• sample image stack
• sample focus
• osa image
• osa focus
• detector image
• generic scan
data: (required) NX_NUMBER

Detectors that provide more than one value per scan point should be summarised
to a single value per scan point for this array in order to simplify plotting.

Note that ‘Line scans’ and focus type scans measure along one spatial dimension but are not restricted to being parallel to the X or Y axes. Such scans should therefore use a single dimension for the positions along the spatial line. The ‘sample_x’ and ‘sample_y’ fields should then contain lists of the x- and y-positions and should both have the ‘axis’ attribute pointing to the same dimension.

energy: (required) NX_FLOAT (Rank: 1, Dimensions: [nE])

List of photon energies of the X-ray beam. If scanned through multiple values,
then an ‘axis’ attribute will be required to link the field to the appropriate data array dimension.

sample_y: (required) NX_FLOAT (Rank: 1, Dimensions: [nY])

List of Y positions on the sample. If scanned through multiple values,
then an ‘axis’ attribute will be required to link the field to the appropriate data array dimension.

sample_x: (required) NX_FLOAT (Rank: 1, Dimensions: [nX])

List of X positions on the sample. If scanned through multiple values,
then an ‘axis’ attribute will be required to link the field to the appropriate data array dimension.

control: (optional) NXmonitor
data: (required) NX_FLOAT
Values to use to normalise for time-variations in photon flux. Typically, the synchrotron storage ring electron beam current is used as a proxy for the X-ray beam intensity. Array must have same shape as the NXdata groups.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXstxm/ENTRY-group
- /NXstxm/ENTRY/control-group
- /NXstxm/ENTRY/control/data-field
- /NXstxm/ENTRY/DATA-group
- /NXstxm/ENTRY/DATA/data-field
- /NXstxm/ENTRY/DATA/energy-field
- /NXstxm/ENTRY/DATA/sample_x-field
- /NXstxm/ENTRY/DATA/sample_y-field
- /NXstxm/ENTRY/DATA/stxm_scan_type-field
- /NXstxm/ENTRY/definition-field
- /NXstxm/ENTRY/end_time-field
- /NXstxm/ENTRY/INSTRUMENT-group
- /NXstxm/ENTRY/INSTRUMENT/DETECTOR-group
- /NXstxm/ENTRY/INSTRUMENT/DETECTOR/data-field
- /NXstxm/ENTRY/INSTRUMENT/monochromator-group
- /NXstxm/ENTRY/INSTRUMENT/monochromator/energy-field
- /NXstxm/ENTRY/INSTRUMENT/sample_x-group
- /NXstxm/ENTRY/INSTRUMENT/sample_x/data-field
- /NXstxm/ENTRY/INSTRUMENT/sample_y-group
- /NXstxm/ENTRY/INSTRUMENT/sample_y/data-field
- /NXstxm/ENTRY/INSTRUMENT/sample_z-group
- /NXstxm/ENTRY/INSTRUMENT/sample_z/data-field
- /NXstxm/ENTRY/INSTRUMENT/SOURCE-group
- /NXstxm/ENTRY/INSTRUMENT/SOURCE/name-field
- /NXstxm/ENTRY/INSTRUMENT/SOURCE/probe-field
- /NXstxm/ENTRY/INSTRUMENT/SOURCE/type-field
- /NXstxm/ENTRY/SAMPLE-group
- /NXstxm/ENTRY/SAMPLE/rotation_angle-field
- /NXstxm/ENTRY/start_time-field
- /NXstxm/ENTRY/title-field
NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXstxm.nxdl.xml

NXtas

Status:
application definition, extends NXobject

Description:
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.

Symbols:
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:
NXcrystal, NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample, NXsource

Structure:

entry: (required) NXentry
  title: (required) NX_CHAR
  start_time: (required) NX_DATE_TIME
  definition: (required) NX_CHAR
    Official NeXus NXDL schema to which this file conforms
    Obligatory value: NXtas

INSTRUMENT: (required) NXinstrument
  SOURCE: (required) NXsource
    name: (required) NX_CHAR
    probe: (required) NX_CHAR
      Any of these values: neutron | x-ray
  monochromator: (required) NXcrystal
    ei: (required) NX_FLOAT (Rank: 1, Dimensions: [nP]) {units=NX_ENERGY}
    rotation_angle: (required) NX_FLOAT (Rank: 1, Dimensions: [nP]) {units=NX_ANGLE}
  analyser: (required) NXcrystal
    ef: (required) NX_FLOAT (Rank: 1, Dimensions: [nP]) {units=NX_ENERGY}
    rotation_angle: (required) NX_FLOAT (Rank: 1, Dimensions: [nP]) {units=NX_ANGLE}
    polar_angle: (required) NX_FLOAT (Rank: 1, Dimensions: [nP]) {units=NX_ANGLE}
**DETECTOR**: (required) `NXdetector`  
  **data**: (required) `NX_INT` (Rank: 1, Dimensions: [nP])  
  **polar_angle**: (required) `NX_FLOAT` (Rank: 1, Dimensions: [nP])  
  `{units=NX_ANGLE}`

**SAMPLE**: (required) `NXsample`  
  **name**: (required) `NX_CHAR`  
  Descriptive name of sample  
  **qh**: (required) `NX_FLOAT` (Rank: 1, Dimensions: [nP])  
  `{units=NX_ANGLE}`  
  **qk**: (required) `NX_FLOAT` (Rank: 1, Dimensions: [nP])  
  `{units=NX_ANGLE}`  
  **ql**: (required) `NX_FLOAT` (Rank: 1, Dimensions: [nP])  
  `{units=NX_ANGLE}`  
  **en**: (required) `NX_FLOAT` (Rank: 1, Dimensions: [nP])  
  `{units=NX_ENERGY}`  
  **rotation_angle**: (required) `NX_FLOAT` (Rank: 1, Dimensions: [nP])  
  `{units=NX_ANGLE}`  
  **polar_angle**: (required) `NX_FLOAT` (Rank: 1, Dimensions: [nP])  
  `{units=NX_ANGLE}`  
  **sgu**: (required) `NX_FLOAT` (Rank: 1, Dimensions: [nP])  
  `{units=NX_ANGLE}`  
  **sgl**: (required) `NX_FLOAT` (Rank: 1, Dimensions: [nP])  
  `{units=NX_ANGLE}`  
  **unit_cell**: (required) `NX_FLOAT` (Rank: 1, Dimensions: [6])  
  `{units=NX_LENGTH}`  
  **orientation_matrix**: (required) `NX_FLOAT` (Rank: 1, Dimensions: [9])  
  `{units=NX_DIMENSIONLESS}`

**MONITOR**: (required) `NXmonitor`  
  **mode**: (required) `NX_CHAR`  
  Count to a preset value based on either clock time (timer) or received monitor counts (monitor).  
  Any of these values: `monitor` | `timer`  
  **preset**: (required) `NX_FLOAT`  
  preset value for time or monitor  
  **data**: (required) `NX_FLOAT` (Rank: 1, Dimensions: [nP])  
  `{units=NX_ANY}`  
  Total integral monitor counts

**DATA**: (required) `NXdata`  
  One of the `ei,ef,qh,qk,ql,en` should get a `primary=1` attribute to denote the main scan axis  
  **ei**: `link` (suggested target: `/NXentry/NXinstrument/monochromator:NXcrystal/ei`)  
  **ef**: `link` (suggested target: `/NXentry/NXinstrument/analyser:NXcrystal/ef`)  
  **en**: `link` (suggested target: `/NXentry/NXsample/en`)
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXtas/entry-group
- /NXtas/entry/DATA-group
- /NXtas/entry/DATA/data-link
- /NXtas/entry/DATA/ef-link
- /NXtas/entry/DATA/ei-link
- /NXtas/entry/DATA/en-link
- /NXtas/entry/DATA/qh-link
- /NXtas/entry/DATA/qk-link
- /NXtas/entry/DATA/ql-link
- /NXtas/entry/definition-field
- /NXtas/entry/INSTRUMENT-group
- /NXtas/entry/INSTRUMENT/analyser-group
- /NXtas/entry/INSTRUMENT/analyser/ef-field
- /NXtas/entry/INSTRUMENT/analyser/polar_angle-field
- /NXtas/entry/INSTRUMENT/analyser/rotation_angle-field
- /NXtas/entry/INSTRUMENT/DETECTOR-group
- /NXtas/entry/INSTRUMENT/DETECTOR/data-field
- /NXtas/entry/INSTRUMENT/DETECTOR/polar_angle-field
- /NXtas/entry/INSTRUMENT/monochromator-group
- /NXtas/entry/INSTRUMENT/monochromator/ei-field
- /NXtas/entry/INSTRUMENT/monochromator/rotation_angle-field
- /NXtas/entry/INSTRUMENT/SOURCE-group
- /NXtas/entry/INSTRUMENT/SOURCE/name-field
- /NXtas/entry/INSTRUMENT/SOURCE/probe-field
- /NXtas/entry/MONITOR-group
- /NXtas/entry/MONITOR/data-field
- /NXtas/entry/MONITOR/mode-field
- /NXtas/entry/MONITOR/preset-field
- /NXtas/entry/SAMPLE-group
• /NXtas/entry/SAMPLE/en-field
• /NXtas/entry/SAMPLE/name-field
• /NXtas/entry/SAMPLE/orientation_matrix-field
• /NXtas/entry/SAMPLE/polar_angle-field
• /NXtas/entry/SAMPLE/qh-field
• /NXtas/entry/SAMPLE/qk-field
• /NXtas/entry/SAMPLE/ql-field
• /NXtas/entry/SAMPLE/rotation_angle-field
• /NXtas/entry/SAMPLE/sgl-field
• /NXtas/entry/SAMPLE/sgu-field
• /NXtas/entry/SAMPLE/unit_cell-field
• /NXtas/entry/start_time-field
• /NXtas/entry/title-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXtas.nxdl.xml

NXtofnpd

Status:
application definition, extends NXobject

Description:
This is a application definition for raw data from a TOF neutron powder diffractometer

Symbols:
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nDet: Number of detectors
nTimeChan: nTimeChan description

Groups cited:
NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample, NXuser

Structure:

  entry: (required) NXentry
    title: (required) NX_CHAR
    start_time: (required) NX_DATE_TIME
    definition: (required) NX_CHAR
    
    Official NeXus NXDL schema to which this file conforms
    Obligatory value: NXtofnpd

    pre_sample_flightpath: (required) NX_FLOAT {units=NX_LENGTH}
This is the flight path before the sample position. This can be determined by a chopper, by the moderator or the source itself. In other words: it the distance to the component which gives the T0 signal to the detector electronics. If another component in the NXinstrument hierarchy provides this information, this should be a link.

**user**: (required) *NXuser*

**name**: (required) *NX_CHAR*

**INSTRUMENT**: (required) *NXinstrument*

**detector**: (required) *NXdetector*

**data**: (required) *NX_INT* (Rank: 2, Dimensions: [nDet, nTimeChan])

**detector_number**: (required) *NX_INT* (Rank: 1, Dimensions: [nDet])

**distance**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet])

{units=*NX_LENGTH*}

distance to sample for each detector

**time_of_flight**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nTimeChan])

{units=*NX_TIME_OF_FLIGHT*}

time_of_flight for each detector element

**polar_angle**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet])

{units=*NX_ANGLE*}

polar angle for each detector element

**azimuthal_angle**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet])

{units=*NX_ANGLE*}

azimuthal angle for each detector element

**SAMPLE**: (required) *NXsample*

**name**: (required) *NX_CHAR*

Descriptive name of sample

**MONITOR**: (required) *NXmonitor*

**mode**: (required) *NX_CHAR*

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: monitor | timer

**preset**: (required) *NX_FLOAT*

preset value for time or monitor

**distance**: (required) *NX_FLOAT* {units=*NX_LENGTH*}

**data**: (required) *NX_INT* (Rank: 1, Dimensions: [nTimeChan])

**time_of_flight**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nTimeChan])

{units=*NX_TIME_OF_FLIGHT*}

data: (required) *NXdata*

**data**: *link* (suggested target: /NXentry/NXinstrument/NXdetector/data)

**detector_number**: *link* (suggested target: /NXentry/NXinstrument/NXdetector/detector_number)
time_of_flight: link (suggested target: /NXentry/NXinstrument/NXdetector/
time_of_flight)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXtofnpd/entry-group
- /NXtofnpd/entry/data-group
- /NXtofnpd/entry/data/data-link
- /NXtofnpd/entry/data/detector_number-link
- /NXtofnpd/entry/data/time_of_flight-link
- /NXtofnpd/entry/definition-field
- /NXtofnpd/entry/INSTRUMENT-group
- /NXtofnpd/entry/INSTRUMENT/detector-group
- /NXtofnpd/entry/INSTRUMENT/detector/azimuthal_angle-field
- /NXtofnpd/entry/INSTRUMENT/detector/data-field
- /NXtofnpd/entry/INSTRUMENT/detector/detector_number-field
- /NXtofnpd/entry/INSTRUMENT/detector/distance-field
- /NXtofnpd/entry/INSTRUMENT/detector/polar_angle-field
- /NXtofnpd/entry/INSTRUMENT/detector/time_of_flight-field
- /NXtofnpd/entry/MONITOR-group
- /NXtofnpd/entry/MONITOR/data-field
- /NXtofnpd/entry/MONITOR/distance-field
- /NXtofnpd/entry/MONITOR/mode-field
- /NXtofnpd/entry/MONITOR/preset-field
- /NXtofnpd/entry/MONITOR/time_of_flight-field
- /NXtofnpd/entry/pre_sample_flightpath-field
- /NXtofnpd/entry/SAMPLE-group
- /NXtofnpd/entry/SAMPLE/name-field
- /NXtofnpd/entry/start_time-field
- /NXtofnpd/entry/title-field
- /NXtofnpd/entry/user-group
- /NXtofnpd/entry/user/name-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXtofnpd.nxdl.xml
**NXtofraw**

**Status:**
application definition, extends *NXobject*

**Description:**
This is an application definition for raw data from a generic TOF instrument

**Symbols:**
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

- **nDet**: Number of detectors
- **nTimeChan**: nTimeChan description

**Groups cited:**
- *NXdata*, *NXdetector*, *NXentry*, *NXinstrument*, *NXmonitor*, *NXsample*, *NXuser*

**Structure:**
- **entry**: (required) *NXentry*
  - **title**: (required) *NX_CHAR*
  - **start_time**: (required) *NX_DATE_TIME*
  - **definition**: (required) *NX_CHAR*
    Official NeXus NXDL schema to which this file conforms
    Obligatory value: NXtofraw
  - **duration**: (required) *NX_FLOAT*
  - **run_number**: (required) *NX_INT*
  - **pre_sample_flightpath**: (required) *NX_FLOAT* {units=*NX_LENGTH*}
    This is the flight path before the sample position. This can be determined by a chopper, by the moderator, or the source itself. In other words: it is the distance to the component which gives the T0 signal to the detector electronics. If another component in the NXinstrument hierarchy provides this information, this should be a link.
  - **user**: (required) *NXuser*
    - **name**: (required) *NX_CHAR*
  - **instrument**: (required) *NXinstrument*
  - **detector**: (required) *NXdetector*
    - **data**: (required) *NX_INT* (Rank: 2, Dimensions: [nDet, nTimeChan])
    - **detector_number**: (required) *NX_INT* (Rank: 1, Dimensions: [nDet])
    - **distance**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_LENGTH*}
      distance to sample for each detector
    - **time_of_flight**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nTimeChan]) {units=*NX_TIME_OF_FLIGHT*}
    - **polar_angle**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_ANGLE*}

---

3.3. NeXus Class Definitions
polar angle for each detector element

**azimuthal_angle**: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nDet])
{units=**NX_ANGLE**}

azimuthal angle for each detector element

**SAMPLE**: (required) **NXSample**

**name**: (required) **NX_CHAR**

Descriptive name of sample

**nature**: (required) **NX_CHAR**

Any of these values: **powder**|**liquid**|**single crystal**

**MONITOR**: (required) **NXmonitor**

**mode**: (required) **NX_CHAR**

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: **monitor**|**timer**

**preset**: (required) **NX_FLOAT**

preset value for time or monitor

**distance**: (required) **NX_FLOAT** {units=**NX_LENGTH**}

**data**: (required) **NX_INT** (Rank: 1, Dimensions: [nTimeChan])

**time_of_flight**: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nTimeChan])
{units=**NX_TIME_OF_FLIGHT**}

**integral_counts**: (required) **NX_INT** {units=**NX_UNITLESS**}

**data**: (required) **NXdata**

**data**: **link** (suggested target: /NXentry/NXinstrument/NXdetector/data)

**detector_number**: **link** (suggested target: /NXentry/NXinstrument/NXdetector/detector_number)

**time_of_flight**: **link** (suggested target: /NXentry/NXinstrument/NXdetector/time_of_flight)

---

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXtofraw/entry-group
- /NXtofraw/entry/data-group
- /NXtofraw/entry/data/data-link
- /NXtofraw/entry/data/detector_number-link
- /NXtofraw/entry/data/time_of_flight-link
- /NXtofraw/entry/definition-field
- /NXtofraw/entry/duration-field
• /NXtofraw/entry/instrument-group
• /NXtofraw/entry/instrument/detector-group
• /NXtofraw/entry/instrument/detector/azimuthal_angle-field
• /NXtofraw/entry/instrument/detector/data-field
• /NXtofraw/entry/instrument/detector/detector_number-field
• /NXtofraw/entry/instrument/detector/distance-field
• /NXtofraw/entry/instrument/detector/polar_angle-field
• /NXtofraw/entry/instrument/detector/time_of_flight-field
• /NXtofraw/entry/MONITOR-group
• /NXtofraw/entry/MONITOR/data-field
• /NXtofraw/entry/MONITOR/distance-field
• /NXtofraw/entry/MONITOR/integral_counts-field
• /NXtofraw/entry/MONITOR/mode-field
• /NXtofraw/entry/MONITOR/preset-field
• /NXtofraw/entry/MONITOR/time_of_flight-field
• /NXtofraw/entry/pre_sample_flightpath-field
• /NXtofraw/entry/run_number-field
• /NXtofraw/entry/SAMPLE-group
• /NXtofraw/entry/SAMPLE/name-field
• /NXtofraw/entry/SAMPLE/nature-field
• /NXtofraw/entry/start_time-field
• /NXtofraw/entry/title-field
• /NXtofraw/entry/user-group
• /NXtofraw/entry/user/name-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXtofraw.nxdl.xml

**NXtofsingle**

**Status:**
application definition, extends **NXobject**

**Description:**
This is a application definition for raw data from a generic TOF instrument

**Symbols:**
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

- **xSize:** xSize description
- **ySize:** ySize description
**nDet**: Number of detectors

**nTimeChan**: nTimeChan description

Groups cited:

NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample, NXuser

Structure:

- **entry**: (required) NXEntry
  - **title**: (required) NX_CHAR
  - **start_time**: (required) NX_DATE_TIME
  - **definition**: (required) NX_CHAR
    - Official NeXus NXDL schema to which this file conforms
    - Obligatory value: NXtofsingle
  - **duration**: (required) NX_FLOAT
  - **pre_sample_flightpath**: (required) NX_FLOAT [units=NX_LENGTH]
    - This is the flight path before the sample position. This can be determined by a chopper, by the moderator or the source itself. In other words: it the distance to the component which gives the T0 signal to the detector electronics. If another component in the NXinstrument hierarchy provides this information, this should be a link.
  - **user**: (required) NXuser
    - **name**: (required) NX_CHAR
  - **INSTRUMENT**: (required) NXinstrument
    - **detector**: (required) NXdetector
      - **data**: (required) NX_INT (Rank: 3, Dimensions: [xSize, ySize, nTimeChan])
      - **distance**: (required) NX_FLOAT (Rank: 1, Dimensions: [1]) [units=NX_LENGTH]
        - Distance to sample for the center of the detector
      - **time_of_flight**: (required) NX_FLOAT (Rank: 1, Dimensions: [nTimeChan]) [units=NX_TIME_OF_FLIGHT]
      - **polar_angle**: (required) NX_FLOAT (Rank: 1, Dimensions: [nDet]) [units=NX_ANGLE]
        - polar angle for each detector element
      - **azimuthal_angle**: (required) NX_FLOAT (Rank: 1, Dimensions: [nDet]) [units=NX_ANGLE]
        - azimuthal angle for each detector element
  - **SAMPLE**: (required) NXsample
    - **name**: (required) NX_CHAR
      - Descriptive name of sample
    - **nature**: (required) NX_CHAR
      - Any of these values: powder|liquid|single crystal
MONITOR: (required) \textit{NXmonitor}

\textbf{mode}: (required) \textit{NX_CHAR}

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: \textit{monitor} | \textit{timer}

\textbf{preset}: (required) \textit{NX_FLOAT}

preset value for time or monitor

\textbf{distance}: (required) \textit{NX_FLOAT} \{\textit{units=NX_LENGTH}\}

\textbf{data}: (required) \textit{NX_FLOAT} (Rank: 1, Dimensions: [nTimeChan])

\textbf{time\_of\_flight}: (required) \textit{NX_FLOAT} \{\textit{units=NX_TIME\_OF\_FLIGHT}\}

\textbf{data}: (required) \textit{NXdata}

\textbf{data}: \textit{link} (suggested target: /NXentry/NXinstrument/NXdetector/data)

\textbf{time\_of\_flight}: \textit{link} (suggested target: /NXentry/NXinstrument/NXdetector/time\_of\_flight)

\section*{Hypertext Anchors}

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXtofsingle/entry-group
- /NXtofsingle/entry/data-group
- /NXtofsingle/entry/data/data-link
- /NXtofsingle/entry/data/time\_of\_flight-link
- /NXtofsingle/entry/definition-field
- /NXtofsingle/entry/duration-field
- /NXtofsingle/entry/INSTRUMENT-group
- /NXtofsingle/entry/INSTRUMENT/detector-group
- /NXtofsingle/entry/INSTRUMENT/detector/azimuthal\_angle-field
- /NXtofsingle/entry/INSTRUMENT/detector/data-field
- /NXtofsingle/entry/INSTRUMENT/detector/distance-field
- /NXtofsingle/entry/INSTRUMENT/detector/polar\_angle-field
- /NXtofsingle/entry/INSTRUMENT/detector/time\_of\_flight-field
- /NXtofsingle/entry/MONITOR-group
- /NXtofsingle/entry/MONITOR/data-field
- /NXtofsingle/entry/MONITOR/distance-field
- /NXtofsingle/entry/MONITOR/mode-field
- /NXtofsingle/entry/MONITOR/preset-field
- /NXtofsingle/entry/MONITOR/time\_of\_flight-field
NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXtofsingle.nxdl.xml

NXtomo

Status:
application definition, extends NXobject

Description:
This is the application definition for x-ray or neutron tomography raw data.
In tomography a number of dark field images are measured, some bright field images and, of course the sample. In order to distinguish between them images carry a image_key.

Symbols:
These symbols will be used below to coordinate datasets with the same shape.

nFrames: Number of frames
xSize: Number of pixels in X direction
ySize: Number of pixels in Y direction

Groups cited:
NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample, NXsource

Structure:
entry: (required) NXentry
  title: (optional) NX_CHAR
  start_time: (optional) NX_DATE_TIME
  end_time: (optional) NX_DATE_TIME
  definition: (required) NX_CHAR
    Official NeXus NXDL schema to which this file conforms
    Obligatory value: NXtomo
  instrument: (required) NXinstrument
    SOURCE: (optional) NXsource
type: (optional) **NX_CHAR**

**name**: (optional) **NX_CHAR**

**probe**: (optional) **NX_CHAR**

Any of these values: neutron | x-ray | electron

de**ector**: (required) **NXdetector**

data**: (required) **NX_INT** (Rank: 3, Dimensions: [nFrames, xSize, ySize])

**image_key**: (required) **NX_INT** (Rank: 1, Dimensions: [nFrames])

In order to distinguish between sample projections, dark and flat images, a magic number is recorded per frame. The key is as follows:

- projection = 0
- flat field = 1
- dark field = 2
- invalid = 3

**x_pixel_size**: (optional) **NX_FLOAT** {units=**NX_LENGTH**}

**y_pixel_size**: (optional) **NX_FLOAT** {units=**NX_LENGTH**}

distance**: (optional) **NX_FLOAT** {units=**NX_LENGTH**}

Distance between detector and sample

**x_rotation_axis_pixel_position**: (optional) **NX_FLOAT**

**y_rotation_axis_pixel_position**: (optional) **NX_FLOAT**

**sample**: (required) **NXsample**

**name**: (required) **NX_CHAR**

Descriptive name of sample

**rotation_angle**: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nFrames]) {units=**NX_ANGLE**}

In practice this axis is always aligned along one pixel direction on the detector and usually vertical. There are experiments with horizontal rotation axes, so this would need to be indicated somehow. For now the best way for that is an open question.

**x_translation**: (optional) **NX_FLOAT** (Rank: 1, Dimensions: [nFrames]) {units=**NX_LENGTH**}

**y_translation**: (optional) **NX_FLOAT** (Rank: 1, Dimensions: [nFrames]) {units=**NX_LENGTH**}

**z_translation**: (optional) **NX_FLOAT** (Rank: 1, Dimensions: [nFrames]) {units=**NX_LENGTH**}

**control**: (optional) **NXmonitor**

data**: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nFrames]) {units=**NX_ANY**}

Total integral monitor counts for each measured frame. Allows a to correction for fluctuations in the beam between frames.

data**: (required) **NXdata**

---

3.3. NeXus Class Definitions
**data**: link (suggested target: /NXentry/NXinstrument/detector:NXdetector/data)

**rotation_angle**: link (suggested target: /NXentry/NXsample/rotation_angle)

**image_key**: link (suggested target: /NXentry/NXinstrument/detector:NXdetector/image_key)

---

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXtomo/entry-group
- /NXtomo/entry/control-group
- /NXtomo/entry/control/data-field
- /NXtomo/entry/data-group
- /NXtomo/entry/data/data-link
- /NXtomo/entry/data/image_key-link
- /NXtomo/entry/data/rotation_angle-link
- /NXtomo/entry/definition-field
- /NXtomo/entry/end_time-field
- /NXtomo/entry/instrument-group
- /NXtomo/entry/instrument/detector-group
- /NXtomo/entry/instrument/detector/data-field
- /NXtomo/entry/instrument/detector/distance-field
- /NXtomo/entry/instrument/detector/image_key-field
- /NXtomo/entry/instrument/detector/x_pixel_size-field
- /NXtomo/entry/instrument/detector/x_rotation_axis_pixel_position-field
- /NXtomo/entry/instrument/detector/y_pixel_size-field
- /NXtomo/entry/instrument/detector/y_rotation_axis_pixel_position-field
- /NXtomo/entry/instrument/SOURCE-group
- /NXtomo/entry/instrument/SOURCE/name-field
- /NXtomo/entry/instrument/SOURCE/probe-field
- /NXtomo/entry/instrument/SOURCE/type-field
- /NXtomo/entry/sample-group
- /NXtomo/entry/sample/name-field
- /NXtomo/entry/sample/rotation_angle-field
- /NXtomo/entry/sample/x_translation-field
- /NXtomo/entry/sample/y_translation-field
- /NXtomo/entry/sample/z_translation-field
NXtomophase

Status:

application definition, extends NXobject

Description:

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.

Symbols:

These symbols will be used below to coordinate datasets with the same shape.

nBrightFrames: Number of bright frames
nDarkFrames: Number of dark frames
nSampleFrames: Number of image (sample) frames
nPhase: Number of phase settings
xSize: Number of pixels in X direction
ySize: Number of pixels in Y direction

Groups cited:

NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample, NXsource

Structure:

entry: (required) NXentry

title: (required) NX_CHAR

start_time: (required) NX_DATE_TIME

end_time: (required) NX_DATE_TIME

definition: (required) NX_CHAR

Official NeXus NXDL schema to which this file conforms

Obligatory value: NXtomophase

instrument: (required) NXinstrument

SOURCE: (required) NXsource

type: (required) NX_CHAR

name: (required) NX_CHAR

probe: (required) NX_CHAR

Any of these values: neutron | x-ray | electron
**bright_field**: (required) **NXdetector**

- **data**: (required) **NX_INT** (Rank: 3, Dimensions: [nBrightFrames, xSize, ySize])
- **sequence_number**: (required) **NX_INT** (Rank: 1, Dimensions: [nBrightFrames])

**dark_field**: (required) **NXdetector**

- **data**: (required) **NX_INT** (Rank: 3, Dimensions: [nDarkFrames, xSize, ySize])
- **sequence_number**: (required) **NX_INT** (Rank: 1, Dimensions: [nDarkFrames])

**sample**: (required) **NXdetector**

- **data**: (required) **NX_INT** (Rank: 4, Dimensions: [nSampleFrames, nPhase, xSize, ySize])
- **sequence_number**: (required) **NX_INT** (Rank: 2, Dimensions: [nSampleFrames, nPhase])
- **x_pixel_size**: (required) **NX_FLOAT** {units=**NX_LENGTH**}
- **y_pixel_size**: (required) **NX_FLOAT** {units=**NX_LENGTH**}
- **distance**: (required) **NX_FLOAT** {units=**NX_LENGTH**}
  
  Distance between detector and sample

**sample**: (required) **NXsample**

- **name**: (required) **NX_CHAR**
  
  Descriptive name of sample

- **rotation_angle**: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nSampleFrames])
  
  {units=**NX_ANGLE**}

- **x_translation**: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nSampleFrames])
  
  {units=**NX_LENGTH**}

- **y_translation**: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nSampleFrames])
  
  {units=**NX_LENGTH**}

- **z_translation**: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nSampleFrames])
  
  {units=**NX_LENGTH**}

**control**: (required) **NXmonitor**

- **integral**: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nDarkFrames + nBrightFrames + nSampleFrame])
  
  {units=**NX_ANY**}

  Total integral monitor counts for each measured frame. Allows a correction for fluctuations in the beam between frames.

**data**: (required) **NXdata**

- **data**: *link* (suggested target: /NXentry/NXinstrument/sample:NXdetector/data)
- **rotation_angle**: *link* (suggested target: /NXentry/NXsample/rotation_angle)
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXtomophase/entry-group
- /NXtomophase/entry/control-group
- /NXtomophase/entry/control/integral-field
- /NXtomophase/entry/data-group
- /NXtomophase/entry/data/data-link
- /NXtomophase/entry/data/rotation_angle-link
- /NXtomophase/entry/definition-field
- /NXtomophase/entry/end_time-field
- /NXtomophase/entry/instrument-group
- /NXtomophase/entry/instrument/bright_field-group
- /NXtomophase/entry/instrument/bright_field/data-field
- /NXtomophase/entry/instrument/bright_field/sequence_number-field
- /NXtomophase/entry/instrument/dark_field-group
- /NXtomophase/entry/instrument/dark_field/data-field
- /NXtomophase/entry/instrument/dark_field/sequence_number-field
- /NXtomophase/entry/instrument/sample-group
- /NXtomophase/entry/instrument/sample/data-field
- /NXtomophase/entry/instrument/sample/distance-field
- /NXtomophase/entry/instrument/sample/sequence_number-field
- /NXtomophase/entry/instrument/sample/x_pixel_size-field
- /NXtomophase/entry/instrument/sample/y_pixel_size-field
- /NXtomophase/entry/instrument/SOURCE-group
- /NXtomophase/entry/instrument/SOURCE/name-field
- /NXtomophase/entry/instrument/SOURCE/probe-field
- /NXtomophase/entry/instrument/SOURCE/type-field
- /NXtomophase/entry/sample-group
- /NXtomophase/entry/sample/name-field
- /NXtomophase/entry/sample/rotation_angle-field
- /NXtomophase/entry/sample/x_translation-field
- /NXtomophase/entry/sample/y_translation-field
- /NXtomophase/entry/sample/z_translation-field
- /NXtomophase/entry/start_time-field
- /NXtomophase/entry/title-field
NXDLS Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXtomophase.nxdl.xml

NXtomoproc

Status:
application definition, extends NXobject

Description:
This is an application definition for the final result of a tomography experiment: a 3D construction of some volume of physical properties.

Symbols:
These symbols will be used below to coordinate datasets with the same shape.

\( nX \): Number of voxels in X direction
\( nY \): Number of voxels in Y direction
\( nZ \): Number of voxels in Z direction

Groups cited:
NXdata, NXentry, NXinstrument, NXparameters, NXprocess, NXsample, NXsource

Structure:

**entry**: (required) NXentry

**title**: (required) NX_CHAR

**definition**: (required) NX_CHAR

Official NeXus NXDL schema to which this file conforms

Obligatory value: NXtomoproc

**INSTRUMENT**: (required) NXinstrument

**SOURCE**: (required) NXsource

**type**: (required) NX_CHAR

**name**: (required) NX_CHAR

**probe**: (required) NX_CHAR

Any of these values: neutron | x-ray | electron

**SAMPLE**: (required) NXsample

**name**: (required) NX_CHAR

Descriptive name of sample

**reconstruction**: (required) NXprocess

**program**: (required) NX_CHAR

Name of the program used for reconstruction

**version**: (required) NX_CHAR

Version of the program used

**date**: (required) NX_DATE_TIME
Date and time of reconstruction processing.

**parameters**: (required) *NXparameters*

**raw_file**: (required) *NX_CHAR*

Original raw data file this data was derived from

**data**: (required) *NXdata*

**data**: (required) *NX_INT* (Rank: 3, Dimensions: [nX, nX, nZ])

This is the reconstructed volume. This can be different things. Please indicate in the unit attribute what physical quantity this really is.

**@transform**: (required) *NX_CHAR*

**@offset**: (required) *NX_CHAR*

**@scaling**: (required) *NX_CHAR*

**x**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nX]) {units=*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.

**y**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nY]) {units=*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.

**z**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nZ]) {units=*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.

---

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXtomoproc/entry-group
- /NXtomoproc/entry/data-group
- /NXtomoproc/entry/data/data-field
- /NXtomoproc/entry/data/data@offset-attribute
- /NXtomoproc/entry/data/data@scaling-attribute
- /NXtomoproc/entry/data/data@transform-attribute
- /NXtomoproc/entry/data/x-field
- /NXtomoproc/entry/data/y-field
- /NXtomoproc/entry/data/z-field
- /NXtomoproc/entry/definition-field
- /NXtomoproc/entry/INSTRUMENT-group
- /NXtomoproc/entry/INSTRUMENT/SOURCE-group
- /NXtomoproc/entry/INSTRUMENT/SOURCE/name-field
- /NXtomoproc/entry/INSTRUMENT/SOURCE/probe-field
• /NXtomoproc/entry/INSTRUMENT/SOURCE/type-field
• /NXtomoproc/entry/reconstruction-group
• /NXtomoproc/entry/reconstruction/date-field
• /NXtomoproc/entry/reconstruction/parameters-group
• /NXtomoproc/entry/reconstruction/parameters/raw_file-field
• /NXtomoproc/entry/reconstruction/program-field
• /NXtomoproc/entry/reconstruction/version-field
• /NXtomoproc/entry/SAMPLE-group
• /NXtomoproc/entry/SAMPLE/name-field
• /NXtomoproc/entry/title-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXtomoproc.nxdl.xml

NXxas

Status:
application definition, extends NXobject

Description:
This is an application definition for raw data from an X-ray absorption spectroscopy experiment.
This is essentially a scan on energy versus incoming/ absorbed beam.

Symbols:
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:
NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXmonochromator, NXsample, NXsource

Structure:
ENTRY: (required) NXentry

@entry: (required) NX_CHAR
NeXus convention is to use “entry1”, “entry2”, … for analysis software to locate each entry.

title: (required) NX_CHAR

start_time: (required) NX_DATE_TIME

definition: (required) NX_CHAR
Official NeXus NXDL schema to which this file conforms
Obligatory value: NXxas

INSTRUMENT: (required) NXinstrument

SOURCE: (required) NXsource
type: (required) **NX_CHAR**
name: (required) **NX_CHAR**
probe: (required) **NX_CHAR**

Obligatory value: x-ray

**monochromator**: (required) **NXmonochromator**

**energy**: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nP])

**incoming_beam**: (required) **NXdetector**

**data**: (required) **NX_NUMBER** (Rank: 1, Dimensions: [nP])

**absorbed_beam**: (required) **NXdetector**

**data**: (required) **NX_NUMBER** (Rank: 1, Dimensions: [nP])

This data corresponds to the sample signal.

**SAMPLE**: (required) **NXsample**

**name**: (required) **NX_CHAR**

Descriptive name of sample

**MONITOR**: (required) **NXmonitor**

**mode**: (required) **NX_CHAR**

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: monitor | timer

**preset**: (required) **NX_FLOAT**

preset value for time or monitor

**data**: (required) **NX_NUMBER** (Rank: 1, Dimensions: [nP])

This field could be a link to /NXentry/NXinstrument/
incoming_beam:NXdetector/data

**DATA**: (required) **NXdata**

**mode**: (required) **NX_CHAR**

Detection method used for observing the sample absorption (pick one from the enumerated list and spell exactly)

Any of these values:

- Total Electron Yield
- Partial Electron Yield
- Auger Electron Yield
- Fluorescence Yield
- Transmission

**energy**: link (suggested target: /NXentry/NXinstrument/monochromator:NXmonochromator/energy)

**absorbed Beam**: link (suggested target: /NXentry/NXinstrument/absorbed_beam:NXdetector/data)
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXxas/ENTRY-group`
- `/NXxas/ENTRY/DATA-group`
- `/NXxas/ENTRY/DATA/absorbed_beam-link`
- `/NXxas/ENTRY/DATA/energy-link`
- `/NXxas/ENTRY/DATA/mode-field`
- `/NXxas/ENTRY/definition-field`
- `/NXxas/ENTRY/INSTRUMENT-group`
- `/NXxas/ENTRY/INSTRUMENT/absorbed_beam-group`
- `/NXxas/ENTRY/INSTRUMENT/absorbed_beam/data-field`
- `/NXxas/ENTRY/INSTRUMENT/incoming_beam-group`
- `/NXxas/ENTRY/INSTRUMENT/incoming_beam/data-field`
- `/NXxas/ENTRY/INSTRUMENT/monochromator-group`
- `/NXxas/ENTRY/INSTRUMENT/monochromator/energy-field`
- `/NXxas/ENTRY/INSTRUMENT/SOURCE-group`
- `/NXxas/ENTRY/INSTRUMENT/SOURCE/name-field`
- `/NXxas/ENTRY/INSTRUMENT/SOURCE/probe-field`
- `/NXxas/ENTRY/INSTRUMENT/SOURCE/type-field`
- `/NXxas/ENTRY/MONITOR-group`
- `/NXxas/ENTRY/MONITOR/data-field`
- `/NXxas/ENTRY/MONITOR/mode-field`
- `/NXxas/ENTRY/MONITOR/preset-field`
- `/NXxas/ENTRY/SAMPLE-group`
- `/NXxas/ENTRY/SAMPLE/name-field`
- `/NXxas/ENTRY/SAMPLE/start_time-field`
- `/NXxas/ENTRY/title-field`
- `/NXxas/ENTRY@entry-attribute`

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxas.nxdl.xml
NXxasproc

Status:
application definition, extends NXobject

Description:
Processed data from XAS. This is energy versus I(incoming)/I(absorbed).

Symbols:
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:
NXdata, NXentry, NXparameters, NXprocess, NXsample

Structure:
ENTRY: (required) NXentry

@entry: (required) NX_CHAR
NeXus convention is to use “entry1”, “entry2”, … for analysis software to locate each entry.

title: (required) NX_CHAR

definition: (required) NX_CHAR
Official NeXus NXDL schema to which this file conforms

Obligatory value: NXxasproc

SAMPLE: (required) NXsample

name: (required) NX_CHAR
Descriptive name of sample

XAS_data_reduction: (required) NXprocess

program: (required) NX_CHAR
Name of the program used for reconstruction

version: (required) NX_CHAR
Version of the program used

date: (required) NX_DATE_TIME
Date and time of reconstruction processing.

parameters: (required) NXparameters

raw_file: (required) NX_CHAR
Original raw data file this data was derived from

DATA: (required) NXdata

energy: (required) NX_CHAR (Rank: 1, Dimensions: [nP])
data: (required) NX_FLOAT (Rank: 1, Dimensions: [nP])
This is corrected and calibrated I(incoming)/I(absorbed). So it is the absorption. Expect attribute signal=1
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXxasproc/ENTRY-group
- /NXxasproc/ENTRY/DATA-group
- /NXxasproc/ENTRY/DATA/data-field
- /NXxasproc/ENTRY/DATA/energy-field
- /NXxasproc/ENTRY/definition-field
- /NXxasproc/ENTRY/SAMPLE-group
- /NXxasproc/ENTRY/SAMPLE/name-field
- /NXxasproc/ENTRY/title-field
- /NXxasproc/ENTRY/XAS_data_reduction-group
- /NXxasproc/ENTRY/XAS_data_reduction/date-field
- /NXxasproc/ENTRY/XAS_data_reduction/parameters-group
- /NXxasproc/ENTRY/XAS_data_reduction/parameters/raw_file-field
- /NXxasproc/ENTRY/XAS_data_reduction/program-field
- /NXxasproc/ENTRY/XAS_data_reduction/version-field
- /NXxasproc/ENTRY@entry-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxasproc.nxdl.xml

NXxbase

Status:
application definition, extends NXobject

Description:
This definition covers the common parts of all monochromatic single crystal raw data application definitions.

Symbols:
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points
nXPixels: Number of X pixels
nYPixels: Number of Y pixels

Groups cited:
NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXmonochromator, NXsample, NXsource

Structure:
entry: (required) NXentry
title: (required) `NX_CHAR`

start_time: (required) `NX_DATE_TIME`

definition: (required) `NX_CHAR`

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXxbase`

instrument: (required) `NXinstrument`

source: (required) `NXsource`

type: (required) `NX_CHAR`

name: (required) `NX_CHAR`

probe: (required) `NX_CHAR`

Any of these values: `neutron` | `x-ray` | `electron`

monochromator: (required) `NXmonochromator`

wavelength: (required) `NX_FLOAT` {`units=NX_WAVELENGTH`}

detector: (required) `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.

data: (required) `NX_INT` (Rank: 3, Dimensions: [nP, nXPixels, nYPixels])

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.

@signal: (required) `NX_POSINT`

Obligatory value: 1

x_pixel_size: (required) `NX_FLOAT` {`units=NX_LENGTH`}

y_pixel_size: (required) `NX_FLOAT` {`units=NX_LENGTH`}

distance: (required) `NX_FLOAT` {`units=NX_LENGTH`}

frame_start_number: (required) `NX_INT`

This is the start number of the first frame of a scan. In PX one often scans a couple of frames on a give sample, then does something else, then returns to the same sample and scans some more frames. Each time with a new data file. This number helps concatenating such measurements.

sample: (required) `NXsample`

name: (required) `NX_CHAR`

Descriptive name of sample

orientation_matrix: (required) `NX_FLOAT` (Rank: 2, Dimensions: [3, 3])

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 the UB nearly always.

unit_cell: (required) `NX_FLOAT` (Rank: 1, Dimensions: [6])
The unit cell, a, b, c, alpha, beta, gamma. Again, not strictly necessary, but normally written.

**temperature**: (required) NX_FLOAT (Rank: 1, Dimensions: [nP])

The sample temperature or whatever sensor represents this value best

**x_translation**: (required) NX_FLOAT {units=NX_LENGTH}

Translation of the sample along the X-direction of the laboratory coordinate system

**y_translation**: (required) NX_FLOAT {units=NX_LENGTH}

Translation of the sample along the Y-direction of the laboratory coordinate system

**distance**: (required) NX_FLOAT {units=NX_LENGTH}

Translation of the sample along the Z-direction of the laboratory coordinate system

**control**: (required) NXmonitor

**mode**: (required) NX_CHAR

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: monitor | timer

**preset**: (required) NX_FLOAT

Preset value for time or monitor

**integral**: (required) NX_FLOAT {units=NX_ANY}

Total integral monitor counts

**DATA**: (required) 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 point to the corresponding detector groups in the instrument hierarchy.

**data**: link (suggested target: /NXentry/NXinstrument/NXdetector/data)

### Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXbase/entry-group
- /NXbase/entry/control-group
- /NXbase/entry/control/integral-field
- /NXbase/entry/control/mode-field
- /NXbase/entry/control/preset-field
- /NXbase/entry/DATA-group
- /NXbase/entry/DATA/data-link
- /NXbase/entry/definition-field
• /NXxbase/entry/instrument-group
• /NXxbase/entry/instrument/detector-group
• /NXxbase/entry/instrument/detector/data-field
• /NXxbase/entry/instrument/detector/data@signal-attribute
• /NXxbase/entry/instrument/detector/distance-field
• /NXxbase/entry/instrument/detector/frame_start_number-field
• /NXxbase/entry/instrument/detector/x_pixel_size-field
• /NXxbase/entry/instrument/detector/y_pixel_size-field
• /NXxbase/entry/instrument/monochromator-group
• /NXxbase/entry/instrument/monochromator/wavelength-field
• /NXxbase/entry/instrument/source-group
• /NXxbase/entry/instrument/source/name-field
• /NXxbase/entry/instrument/source/probe-field
• /NXxbase/entry/instrument/source/type-field
• /NXxbase/entry/sample-group
• /NXxbase/entry/sample/distance-field
• /NXxbase/entry/sample/name-field
• /NXxbase/entry/sample/orientation_matrix-field
• /NXxbase/entry/sample/temperature-field
• /NXxbase/entry/sample/unit_cell-field
• /NXxbase/entry/sample/x_translation-field
• /NXxbase/entry/sample/y_translation-field
• /NXxbase/entry/start_time-field
• /NXxbase/entry/title-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxbase.nxdl.xml

NXxeuler

Status:
application definition, extends NXxbase

Description:
raw data from a four-circle diffractometer with an eulerian cradle, extends NXxbase

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 arbitrary scans in reciprocal space.

Symbols:
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points
Groups cited:
    NXdata, NXdetector, NXentry, NXinstrument, NXsample

Structure:

**entry**: (required) **NXEntry**

  **definition**: (required) **NX_CHAR**

  Official NeXus NXDL schema to which this file conforms

  Obligatory value: **NXxeuler**

**instrument**: (required) **NXinstrument**

**detector**: (required) **NXdetector**

  **polar_angle**: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nP])
  {units=**NX_ANGLE**}

  The polar_angle (two theta) where the detector is placed at each scan point.

**sample**: (required) **NXsample**

  **rotation_angle**: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nP])
  {units=**NX_ANGLE**}

  This is an array holding the sample rotation angle at each scan point

  **chi**: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nP])
  {units=**NX_ANGLE**}

  This is an array holding the chi angle of the eulerian cradle at each scan point

  **phi**: (required) **NX_FLOAT** (Rank: 1, Dimensions: [nP])
  {units=**NX_ANGLE**}

  This is an array holding the phi rotation of the eulerian cradle at each scan point

**name**: (required) **NXdata**

  **polar_angle**:  link (suggested target: /NXentry/NXinstrument/NXdetector/
  polar_angle)

  **rotation_angle**:  link (suggested target: /NXentry/NXsample/rotation_angle)

  **chi**:  link (suggested target: /NXentry/NXsample/chi)

  **phi**:  link (suggested target: /NXentry/NXsample/phi)

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXxeuler/entry-group
- /NXxeuler/entry/definition-field
- /NXxeuler/entry/instrument-group
- /NXxeuler/entry/instrument/detector-group
- /NXxeuler/entry/instrument/detector/polar_angle-field
- /NXxeuler/entry/name-group
- /NXxeuler/entry/name/chi-link
NXkappa

Status:
application definition, extends NXbase

Description:
raw data from a kappa geometry (CAD4) single crystal diffractometer, extends NXbase

This is the application definition for raw data from a kappa geometry (CAD4) single crystal diffractometer. It extends NXbase, so the full definition is the content of NXbase plus the data defined here.

Symbols:
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:
NXdata, NXdetector, NXentry, NXinstrument, NXsample

Structure:

entry: (required) NXentry

definition: (required) NX_CHAR

Official NeXus NXDL schema to which this file conforms

Obligatory value: NXkappa

instrument: (required) NXinstrument
detector: (required) NXdetector

polar_angle: (required) NX_FLOAT (Rank: 1, Dimensions: [nP]) {units=NX_ANGLE}

The polar_angle (two theta) at each scan point

sample: (required) NXsample

rotation_angle: (required) NX_FLOAT (Rank: 1, Dimensions: [nP]) {units=NX_ANGLE}

This is an array holding the sample rotation angle at each scan point

kappa: (required) NX_FLOAT (Rank: 1, Dimensions: [nP]) {units=NX_ANGLE}

This is an array holding the kappa angle at each scan point
**phi**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*}

This is an array holding the phi angle at each scan point

**alpha**: (required) *NX_FLOAT* {units=*NX_ANGLE*}

This holds the inclination angle of the kappa arm.

**name**: (required) *NXdata*

- **polar_angle**: *link* (suggested target: /NXentry/NXinstrument/NXdetector/polar_angle)
- **rotation_angle**: *link* (suggested target: /NXentry/NXsample/rotation_angle)
- **kappa**: *link* (suggested target: /NXentry/NXsample/kappa)
- **phi**: *link* (suggested target: /NXentry/NXsample/phi)

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXkappa/entry-group
- /NXkappa/entry/definition-field
- /NXkappa/entry/instrument-group
- /NXkappa/entry/instrument/detector-group
- /NXkappa/entry/instrument/detector/polar_angle-field
- /NXkappa/entry/name-group
- /NXkappa/entry/name/kappa-link
- /NXkappa/entry/name/phi-link
- /NXkappa/entry/name/polar_angle-link
- /NXkappa/entry/name/rotation_angle-link
- /NXkappa/entry/sample-group
- /NXkappa/entry/sample/alpha-field
- /NXkappa/entry/sample/kappa-field
- /NXkappa/entry/sample/phi-field
- /NXkappa/entry/sample/rotation_angle-field

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/applications/NXkappa.nxdl.xml
**NXxlaue**

**Status:**
application definition, extends NXxrot

**Description:**
raw data from a single crystal laue camera, extends NXxrot

This is the application definition for raw data from a single crystal laue camera. It extends NXxrot.

**Symbols:**
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nE: Number of energies

**Groups cited:**
NXdata, NXentry, NXinstrument, NXsource

**Structure:**
entry: (required) NXentry
  definition: (required) NX_CHAR
    Official NeXus NXDL schema to which this file conforms
    Obligatory value: NXxlaue
  instrument: (required) NXinstrument
  source: (required) NXsource
    distribution: (required) NXdata
      This is the wavelength distribution of the beam
      data: (required) NX_CHAR (Rank: 1, Dimensions: [nE])
        expect signal=1 axes="energy"
      wavelength: (required) NX_CHAR (Rank: 1, Dimensions: [nE])
        {units=NX_WAVELENGTH}

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXxlaue/entry-group
- /NXxlaue/entry/definition-field
- /NXxlaue/entry/instrument-group
- /NXxlaue/entry/instrument/source-group
- /NXxlaue/entry/instrument/source/distribution-group
- /NXxlaue/entry/instrument/source/distribution/data-field
- /NXxlaue/entry/instrument/source/distribution/wavelength-field

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/applications/NXxlaue.nxdl.xml

3.3. NeXus Class Definitions
NXxlaueplate

Status:

application definition, extends *NXxlaue*

Description:

raw data from a single crystal Laue camera, extends *NXxlaue*

This is the application definition for raw data from a single crystal Laue camera with an image plate as a detector. It extends *NXxlaue*.

Symbols:

No symbol table

Groups cited:

*NXdetector*, *NXentry*, *NXinstrument*

Structure:

entry: (required) *NXentry*

  definition: (required) *NX_CHAR*

    Official NeXus NXDL schema to which this file conforms

    Obligatory value: *NXxlaueplate*

instrument: (required) *NXinstrument*

detector: (required) *NXdetector*

  diameter: (required) *NX_FLOAT* {units=*NX_LENGTH*}

    The diameter of a cylindrical detector

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXxlaueplate/entry-group
- /NXxlaueplate/entry/definition-field
- /NXxlaueplate/entry/instrument-group
- /NXxlaueplate/entry/instrument/detector-group
- /NXxlaueplate/entry/instrument/detector/diameter-field

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/applications/NXxlaueplate.nxdl.xml
NXxn

Status:

application definition, extends NXxb

Description:

raw data from a single crystal diffractometer, extends NXxb

This is the application definition for raw data from a single crystal diffractometer measuring in normal beam mode. It extends NXxb, so the full definition is the content of NXxb plus the data defined here. All angles are logged in order to support arbitrary scans in reciprocal space.

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:
NXdata, NXdetector, NXentry, NXinstrument, NXsample

Structure:

entry: (required) NXentry

definition: (required) NX_CHAR

Official NeXus NXDL schema to which this file conforms

Obligatory value: NXxn

instrument: (required) NXinstrument

detector: (required) NXdetector

polar_angle: (required) NX_FLOAT (Rank: 1, Dimensions: [nP])
{units=NX_ANGLE}

The polar_angle (gamma) of the detector for each scan point.

tilt_angle: (required) NX_FLOAT (Rank: 1, Dimensions: [nP])
{units=NX_ANGLE}

The angle by which the detector has been tilted out of the scattering plane.

sample: (required) NXsample

rotation_angle: (required) NX_FLOAT (Rank: 1, Dimensions: [nP])
{units=NX_ANGLE}

This is an array holding the sample rotation angle at each scan point.

name: (required) NXdata

polar_angle: link (suggested target: /NXentry/NXinstrument/NXdetector/polar_angle)

tilt: link (suggested target: /NXentry/NXinstrument/NXdetector/tilt)

rotation_angle: link (suggested target: /NXentry/NXsample/rotation_angle)
**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXxnb/entry-group`
- `/NXxnb/entry/definition-field`
- `/NXxnb/entry/instrument-group`
- `/NXxnb/entry/instrument/detector-group`
- `/NXxnb/entry/instrument/detector/polar_angle-field`
- `/NXxnb/entry/instrument/detector/tilt_angle-field`
- `/NXxnb/entry/name-group`
- `/NXxnb/entry/name/polar_angle-link`
- `/NXxnb/entry/name/rotation_angle-link`
- `/NXxnb/entry/name/tilt-link`
- `/NXxnb/entry/sample-group`
- `/NXxnb/entry/sample/rotation_angle-field`

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/applications/NXxnb.nxdl.xml

**NXxrot**

**Status:**
application definition, extends *NXxbase*

**Description:**
raw data from a rotation camera, extends *NXxbase*

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.

**Symbols:**
The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

**Groups cited:**
*NXattenuator*, *NXdata*, *NXdetector*, *NXentry*, *NXinstrument*, *NXsample*

**Structure:**

- **entry**: (required) *NXentry*

  - **definition**: (required) *NX_CHAR*

    Official NeXus NEXDL schema to which this file conforms.

    Obligatory value: *NXxrot*

  - **instrument**: (required) *NXinstrument*

  - **detector**: (required) *NXdetector*
**polar_angle**: (required) *NX_FLOAT* {units=*NX_ANGLE*}

The polar_angle (two theta) where the detector is placed.

**beam_center_x**: (required) *NX_FLOAT* {units=*NX_LENGTH*}

This is the x position where the direct beam would hit the detector. This is a length, not a pixel position, and can be outside of the actual detector.

**beam_center_y**: (required) *NX_FLOAT* {units=*NX_LENGTH*}

This is the y position where the direct beam would hit the detector. This is a length, not a pixel position, and can be outside of the actual detector.

**attenuator**: (required) *NXattenuator*

**attenuator_transmission**: (required) *NX_FLOAT* {units=*NX_ANY*}

**sample**: (required) *NXsample*

**rotation_angle**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*}

This is an array holding the sample rotation start angle at each scan point.

**rotation_angle_step**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*}

This is an array holding the step made for sample rotation angle at each scan point.

**name**: (required) *NXdata*

**rotation_angle**: link (suggested target: /NXentry/NXsample/rotation_angle)

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXxrot/entry-group
- /NXxrot/entry/definition-field
- /NXxrot/entry/instrument-group
- /NXxrot/entry/instrument/attenuator-group
- /NXxrot/entry/instrument/attenuator/attenuator_transmission-field
- /NXxrot/entry/instrument/detector-group
- /NXxrot/entry/instrument/detector/beam_center_x-field
- /NXxrot/entry/instrument/detector/beam_center_y-field
- /NXxrot/entry/instrument/detector/polar_angle-field
- /NXxrot/entry/name-group
- /NXxrot/entry/name/rotation_angle-link
- /NXxrot/entry/sample-group
- /NXxrot/entry/sample/rotation_angle-link
- /NXxrot/entry/sample/rotation_angle-field
- /NXxrot/entry/sample/rotation_angle_step-field
3.3.3 Contributed Definitions

A description of each NeXus contributed definition is given. NXDL files in the NeXus contributed definitions include propositions from the community for NeXus base classes or application definitions, as well as other NXDL files for long-term archival by NeXus. Consider the contributed definitions as either in incubation or a special case not for general use. The NIAC: The NeXus International Advisory Committee is charged to review any new contributed definitions and provide feedback to the authors before ratification and acceptance as either a base class or application definition.

NXaberration
Models for aberrations of electro-magnetic lenses in electron microscopy.

NXaperture_em
Details of an individual aperture for electron beams.

NXapm
Application definition for atom probe microscopy experiments.

NXcalibration
Subclass of NXprocess to describe post-processing calibrations.

NXchamber
Component of an instrument to store or place objects and specimens.

NXcollectioncolumn
Subclass of NXelectronanalyser to describe the electron collection column of a

NXcontainer
State of a container holding the sample under investigation.

NXcoordinate_system_set
Container to hold different coordinate systems conventions.

NXcorrector_cs
Corrector for aberrations in an electron microscope.

NXcsg
Constructive Solid Geometry base class, using NXquadric and NXoff_geometry

NXcxi_ptycho
Application definition for a ptychography experiment, compatible with CXI from version 1.6.

NXdeflector
Deflectors as they are used e.g. in an electron analyser.

NXdistortion
Subclass of NXprocess to describe post-processing distortion correction.

NXebeam_column
Container for components to form a controlled electron beam.

NXelectronanalyser
Subclass of NXinstrument to describe a photoelectron analyser.

NXelectrostatic_kicker
definition for a electrostatic kicker.

NXellipsometry
Ellipsometry, complex systems, up to variable angle spectroscopy.
NXem
Characterization and session with one sample in an electron microscope.

NXenergydispersion
Subclass of NXelectronanalyser to describe the energy dispersion section of a

NXevent_data_em
Metadata and settings of an electron microscope for scans and images.

NXevent_data_em_set
Container to hold NXevent_data_em instances of an electron microscope session.

NXibeam_column
Container for components of a focused-ion-beam (FIB) system.

NXimage_set_em_adf
Container for reporting a set of annular dark field images.

NXimage_set_em_bf
Container for reporting a set of images taken in bright field mode.

NXimage_set_em_bse
Container for reporting a set of back-scattered electron images.

NXimage_set_em_chamber
Container for images recorded with e.g. a TV camera in the microscope chamber.

NXimage_set_em_df
Container for reporting a set of images taken in dark field mode.

NXimage_set_em_diffrac
Container for reporting a set of diffraction images.

NXimage_set_em_ecci
Container for reporting back-scattered electron channeling contrast images.

NXimage_set_em_kikuchi
Electron backscatter diffraction (EBSD) Kikuchi pattern.

NXimage_set_em_ronchigram
Container for reporting a set of images related to a ronchigram.

NXimage_set_em_se
Container for reporting a set of secondary electron images.

NXinteraction_vol_em
Base class for storing details about a modelled shape of interaction volume.

NXion
Set of atoms of a molecular ion or fragment in e.g. ToF mass spectrometry.

NXlens_em
Description of an electro-magnetic lens or a compound lens.

NXmagnetic_kicker
definition for a magnetic kicker.

NXmanipulator
Extension of NXpositioner to include fields to describe the use of manipulators

NXmanufacturer
Details about a component as defined by its manufacturer.
NXmpes
  This is the most general application definition for multidimensional

NXoptical_system_em
  A container for qualifying an electron optical system.

NXpeak
  Description of peaks, their functional form or measured support.

NXpulser_apm
  Metadata for laser-, voltage-, or combined pulsing triggering field evaporation.

NXpump
  Device to reduce an atmosphere to a controlled remaining pressure level.

NXquadric
  definition of a quadric surface.

NXquadrupole_magnet
  definition for a quadrupole magnet.

NXreflectron
  Device for reducing flight time differences of ions in ToF experiments.

NXregion
  Geometry and logical description of a region of data in a parent group. When used, it could be a child group to, say, NXdetector.

NXregistration
  Describes image registration procedures.

NXscanbox_em
  Scan box and coils which deflect an electron beam in a controlled manner.

NXseparator
  definition for an electrostatic separator.

NXsnsevent
  This is a definition for event data from Spallation Neutron Source (SNS) at ORNL.

NXsnshisto
  This is a definition for histogram data from Spallation Neutron Source (SNS) at ORNL.

NXsolenoid_magnet
  definition for a solenoid magnet.

NXsolid_geometry
  The head node for constructively defined geometry

NXspectrum_set_em_auger
  Container for reporting a set of auger electron energy spectra.

NXspectrum_set_em_cathodolum
  Container for reporting a set of cathodoluminescence spectra.

NXspectrum_set_em_eels
  Container for reporting a set of electron energy loss spectra.

NXspectrum_set_em_xray
  Container for reporting a set of energy-dispersive X-ray spectra.

NXspin_rotator
  definition for a spin rotator.
NXspindispersion
Subclass of NXelectronanalyser to describe the spin filters in photoemission

NXstage_lab
A stage lab can be used to hold, align, orient, and prepare a specimen.

NXxpcs
X-ray Photon Correlation Spectroscopy (XPCS) data (results).

NXaberration

Status:
base class, extends NXobject

Description:
Models for aberrations of electro-magnetic lenses in electron microscopy.

The notation follows O. Krivanek et al. (1999) and O. Krivanek et al. (2003) See also S. J. Pennycock and P. D. Nellist (page 44ff, and page 118ff) for further details, additional literature, and the unit of the coefficients. Consult Table 7-2 of Ibid. publication on how to convert between conventions of different groups/vendors.

Symbols:
No symbol table

Groups cited:
none

Structure:

\( c_{1\_0} \): (optional) \( \text{NX\_FLOAT} \) \( \{ \text{units} = \text{NX\_LENGTH} \} \)

Defocus

\( c_{1\_2\_a} \): (optional) \( \text{NX\_FLOAT} \) \( \{ \text{units} = \text{NX\_LENGTH} \} \)

Two-fold astigmatism

\( c_{1\_2\_b} \): (optional) \( \text{NX\_FLOAT} \) \( \{ \text{units} = \text{NX\_LENGTH} \} \)

Two-fold astigmatism

\( c_{2\_1\_a} \): (optional) \( \text{NX\_FLOAT} \) \( \{ \text{units} = \text{NX\_LENGTH} \} \)

Second-order axial coma

\( c_{2\_1\_b} \): (optional) \( \text{NX\_FLOAT} \) \( \{ \text{units} = \text{NX\_LENGTH} \} \)

Second-order axial coma

\( c_{2\_3\_a} \): (optional) \( \text{NX\_FLOAT} \) \( \{ \text{units} = \text{NX\_LENGTH} \} \)

Threefold astigmatism

\( c_{2\_3\_b} \): (optional) \( \text{NX\_FLOAT} \) \( \{ \text{units} = \text{NX\_LENGTH} \} \)

Threefold astigmatism

\( c_{3\_0} \): (optional) \( \text{NX\_FLOAT} \) \( \{ \text{units} = \text{NX\_LENGTH} \} \)

Spherical aberration

\( c_{3\_2\_a} \): (optional) \( \text{NX\_FLOAT} \) \( \{ \text{units} = \text{NX\_LENGTH} \} \)
Star aberration
c_3_2_b: (optional) NX_FLOAT \{units=NX_LENGTH\}

Star aberration
c_3_4_a: (optional) NX_FLOAT \{units=NX_LENGTH\}

Fourfold astigmatism
c_3_4_b: (optional) NX_FLOAT \{units=NX_LENGTH\}

Fourfold astigmatism
c_5_0: (optional) NX_FLOAT \{units=NX_LENGTH\}

Fifth-order spherical aberration

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXaberration/c_1_0-field
- /NXaberration/c_1_2_a-field
- /NXaberration/c_1_2_b-field
- /NXaberration/c_2_1_a-field
- /NXaberration/c_2_1_b-field
- /NXaberration/c_2_3_a-field
- /NXaberration/c_2_3_b-field
- /NXaberration/c_3_0-field
- /NXaberration/c_3_2_a-field
- /NXaberration/c_3_2_b-field
- /NXaberration/c_3_4_a-field
- /NXaberration/c_3_4_b-field
- /NXaberration/c_5_0-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXaberration.nxdl.xml

NXaperture_em

Status:
base class, extends NXObject

Description:
Details of an individual aperture for electron beams.

Symbols:
No symbol table
Groups cited:

NXmanufacturer, NXtransformations

Structure:

name: (optional) NX_CHAR

Given name/alias of the aperture.

value: (optional) NX_NUMBER {units=NX_ANY }

Relevant value from the control software.

This is not always just the diameter of (not even in the case) of a circular aperture. Usually it is a mode setting value which is selected in the control software. Which settings are behind the value should be defined for now in the description field, if these are known in more detail.

description: (optional) NX_CHAR

Ideally, a (globally) unique persistent identifier, link, or text to a resource which gives further details. Alternatively a free-text field.

MANUFACTURER: (optional) NXmanufacturer

TRANSFORMATIONS: (optional) NXtransformations

Affine transformation which detail the arrangement in the microscope relative to the optical axis and beam path.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

• /NXaperture_em/description-field
• /NXaperture_em/MANUFACTURER-group
• /NXaperture_em/name-field
• /NXaperture_em/TRANSFORMATIONS-group
• /NXaperture_em/value-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXaperture_em.nxdl.xml

NXapm

Status:

application definition, extends NXobject

Description:

Application definition for atom probe microscopy experiments.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays

n_ions: Total number of ions collected

n_dld_wires: Total number of independent wires in the delay-line detector.

n_support: Number of support points for e.g. modeling peaks.
n_ivec_max: Maximum number of allowed atoms per (molecular) ion (fragment). Needs to match maximum_number_of_atoms_per_molecular_ion.

n_ranges: Number of mass-to-charge-state-ratio range intervals mapped on this ion type.

n_x: Number of bins in the x direction.

n_y: Number of bins in the y direction.

n_z: Number of bins in the z direction.

n_bins: Number of bins.

Groups cited:

NXaperture_em, NXchamber, NXcollection, NXcoordinate_system_set, NXdata, NXdetector, NXentry, NXinstrument, NXion, NXlens_em, NXmanufacturer, NXmonitor, NXnote, NXpeak, NXprocess, NXPulser_apm, NXPump, NXreflector, NXsample, NXstage_lab, NXuser

Structure:

ENTRY: (required) NXentry

@version: (required) NX_CHAR

An at least as strong as SHA256 hashvalue of the file that specifies the application definition.

definition: (required) NX_CHAR

NeXus NXDL schema to which this file conforms.

Obligatory value: NXapm

experiment_identifier: (required) NX_CHAR

Ideally, a (globally) unique persistent identifier for referring to this experiment.

The identifier is usually defined/issued by the facility, laboratory, or the principle investigator. The identifier enables to link experiments to e.g. proposals.

experiment_description: (optional) NX_CHAR

Free-text description about the experiment.

Users are strongly advised to detail the sample history in the respective field and fill rather as completely as possible the fields of this application definition rather than write details about the experiment into this free-text description field.

start_time: (required) NX_DATE_TIME

ISO 8601 time code with local time zone offset to UTC information included when the microscope session started. If the application demands that time codes in this section of the application definition should only be used for specifying when the experiment was performed - and the exact duration is not relevant - this start time field should be used.

Often though it is useful to specify a time interval with specifying both start_time and end_time to allow for more detailed bookkeeping and interpretation of the experiment. The user should be aware that even with having both time instances specified, it may not be possible to infer how long the experiment took or for how long data were acquired.

More detailed timing data over the course of the experiment have to be collected to compute this. These computations can take advantage of individual time stamps in NXevent_em instances to provide additional pieces of information.
end_time: (recommended) *NX_DATE_TIME*

ISO 8601 time code with local time zone offset to UTC included when the microscope session ended.

program: (required) *NX_CHAR*

Commercial or otherwise given name to the program which was used to create the file. Atom probe microscopy experiments are nowadays in most cases controlled via commercial software. These are often designed as integrated acquisition and instrument control software solutions. For AMETEK/Cameca local electrode atom probe (LEAP) instruments the least processed (rawest) numerical results and metadata are stored in so-called RHIT and HITS files, which are proprietary and the specifications of which are not publicly documented.

Supplementary metadata are kept in a database which is connected to the instrument control software. RHIT and HITS are proprietary binary file formats whose content must not be accessed with software other than of AMETEK (IVAS/AP Suite). In effect, RHIT and HITS files store the experiment in a closed manner that is practically useless for users unless they have access to the commercial software.

To arrive at a state that atom probe microscopy with LEAP instruments delivers a dataset with which users can study reconstructed atomic position and do e.g. composition analyses or other post-processing analysis tasks, these raw data have to be processed. Therefore, it is necessary that for an application definition to be useful, details about the physical acquisition of the raw data and all its processing steps have to be stored.

With this a user can create derived quantities like ion hit positions (on the detector), calibrated time-of-flight data. These derived quantities are also needed to obtain calibrated mass-to-charge-state ratios, and finally the tomographic reconstruction of the ion positions.

In most cases, an APM dataset is useful only if it gets post-processed via so-called ranging. Ranging defines rules for mapping time-of-flight and mass-to-charge-state ratio values on ion species. In turn, these labels decode elemental identities and can often also be used to resolve isotopes. All these steps are in most cases performed using commercial software.

Frequently, though, ranging and post-processing is also performed with (open-source) research software. Therefore, there is strictly speaking not a single program used throughout an atom probe analysis not even for the early data acquisition and processing stages to obtain a useful reconstructed and ranged dataset.

Therefore, the application definition documents not only the measurement but also the key post-processing steps which transform the proprietary data into a tomographic reconstruction with ranging definitions.

@version: (required) *NX_CHAR*

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

run_number: (required) *NX_CHAR*

Not the specimen name or the experiment identifier but the identifier through which the experiment is referred to in the control software. For LEAP instruments it is recommended to use the IVAS/AP Suite run_number. For other instruments, such as
the one from Stuttgart or Oxcart from Erlangen, or the instruments in Rouen, use the identifier which is closest in meaning to the LEAP run number.

As a destructive microscopy method, a run can be performed only once. It is possible, however, to interrupt a run and restart data acquisition while still using the same specimen. In this case, each evaporation run needs to be distinguished with different run numbers. We follow this habit of most atom probe groups.

**operation_mode**: (required) **NX_CHAR**

What type of atom probe microscope experiment is performed. This field can be used e.g. by materials database systems to qualitatively filter experiments. APT are experiments where the analysis_chamber has no imaging gas. For FIM analyses an imaging gas is used, which should be specified with the atmosphere in the analysis_chamber group. Combinations of the two imaging modes are possible. For these apt_fim or other operation_mode the user should specify details in the experiment_documentation field.

Any of these values: apt | fim | apt_fim | other

**experiment_documentation**: (optional) **NXnote**

Binary container for a file or a compressed collection of files which can be used to add further descriptions and details to the experiment. The container can hold a compressed archive.

Required for operation_mode apt_fim or other to give further details. Users should not abuse this field to provide free-text information. Instead, these should be mapped to respective groups and sections.

**thumbnail**: (recommended) **NXnote**

A small image that is representative of the entry; this can be an image taken from the dataset like a thumbnail of a spectrum. A 640 x 480 pixel jpeg image is recommended. Adding a scale bar to that image is recommended but not required as the main purpose of the thumbnail is to provide e.g. thumbnail images for displaying them in data repositories.

**@type**: (required) **NX_CHAR**

**operator**: (required) **NXuser**

Contact information and eventually details of at least one person involved in the taking of the microscope session. This can be the principle investigator who performed this experiment. Adding multiple users if relevant is recommended.

**name**: (required) **NX_CHAR**

Given (first) name and surname of the user.

**affiliation**: (recommended) **NX_CHAR**

Name of the affiliation of the user at the point in time when the experiment was performed.

**address**: (recommended) **NX_CHAR**

Postal address of the affiliation.

**email**: (required) **NX_CHAR**

Email address of the user at the point in time when the experiment was performed. Writing the most permanently used email is recommended.
orcid: (recommended) **NX_CHAR**

Globally unique identifier of the user as offered by services like ORCID or ResearcherID.

**telephone_number:** (optional) **NX_CHAR**

(Business) (tele)phone number of the user at the point in time when the experiment was performed.

**role:** (optional) **NX_CHAR**

Which role does the user have in the place and at the point in time when the experiment was performed? Technician operating the microscope. Student, postdoc, principle investigator, guest are common examples.

**social_media_name:** (optional) **NX_CHAR**

Account name that is associated with the user in social media platforms.

**social_media_platform:** (optional) **NX_CHAR**

Name of the social media platform where the account under social_media_name is registered.

**specimen:** (required) **NXSample**

**name:** (required) **NX_CHAR**

Descriptive name or ideally (globally) unique persistent identifier. The name distinguishes the specimen from all others and especially the predecessor/origin from where the specimen was cut. In cases where the specimen was e.g. site-specifically cut from samples or in cases of an instrument session during which multiple specimens are loaded, the name has to be descriptive enough to resolve which specimen on e.g. the microtip array was taken.

The user is advised to store the details how specimens were cut/prepared from samples in the sample history. This field must not be used for an alias of the specimen. Instead, use short_title.

In cases where multiple specimens have been loaded into the microscope the name has to be the specific one, whose results are stored by this NXentry, because a single NXentry should be used only for the characterization of a single specimen.

Details about the specimen preparation should be stored in the sample history.

**sample_history:** (required) **NX_CHAR**

Ideally, a reference to the location of or a (globally) unique persistent identifier of e.g. another file which should document ideally as many details as possible of the material, its microstructure, and its thermo-chemo-mechanical processing/preparation history.

In the case that such a detailed history of the sample/specimen is not available, use this field as a free-text description to specify a sub-set of the entire sample history, i.e. what you would consider being the key steps and relevant information about the specimen, its material, microstructure, thermo-chemo-mechanical processing state and details of the preparation.

**preparation_date:** (required) **NX_DATE_TIME**

ISO 8601 time code with local time zone offset to UTC information when the specimen was prepared.
Ideally report the end of the preparation, i.e. the last known time the measured specimen surface was actively prepared. Usually this should be a part of the sample history, i.e. the sample is imagined handed over for the analysis. At the point it enters the microscope the session starts.

Knowing when the specimen was exposed to e.g. specific atmosphere is especially required for environmentally sensitive material such as hydrogen charged specimens or experiments including tracers with a short half time. Further time stamps prior to preparation_date should better be placed in resources which describe the sample_history.

**short_title:** (optional) *NX_CHAR*

Possibility to give an abbreviation of the specimen name field.

**atom_types:** (required) *NX_CHAR*

Use Hill’s system for listing elements of the periodic table which are inside or attached to the surface of the specimen and thus relevant from a scientific point of view.

The purpose of the field is to offer materials database systems an opportunity to parse the relevant elements without having to interpret these from the sample history.

**description:** (optional) *NX_CHAR*

Discouraged free-text field in case properly designed records for the sample_history are not available.

**DATA:** (required) *NXdata*

Hard link to a location in the hierarchy of the NeXus file where the data for default plotting are stored.

**COORDINATE_SYSTEM_SET:** (required) *NXcoordinate_system_set*

Container to hold different coordinate systems conventions.

For the specific idea and conventions to use with the NXcoordinate_system_set inspect the description of the NXcoordinate_system_set base class.

Specific details for application in atom probe microscopy follow. In this research field scientists distinguish usually several Euclidean coordinate systems (CS):

- The laboratory space; a CS specifying the room where the instrument is located in or a physical landmark on the instrument, e.g. the direction of the transfer rod where positive is the direction how the rod pushes into the instrument.
- The specimen space; a CS affixed to either the base or the initial apex of the specimen, whose z axis points towards the detector.
- The detector space; a CS affixed whose xy plane is usually in the detector and whose z axis points towards the specimen.
- The reconstruction space; a CS associated with the tomographic reconstruction. Its orientation depends on the commercial software used.
- Eventually further coordinate systems attached to the flight path of individual ions might be defined.

Coordinate systems should be right-handed ones. Clockwise rotations should be considered positive rotations.
In atom probe microscopy a frequently used choice for the detector space (CS) is discussed with the so-called detector space image (stack). This is a stack of two-dimensional histograms of detected ions within a predefined evaporation ID interval. Typically, the set of ion evaporation sequence IDs is grouped into chunks.

For each chunk a histogram of the ion hit positions on the detector is computed. This leaves the possibility for inconsistency between the so-called detector space and the e.g. specimen space.

The transformation here resolves this ambiguity by specifying how the positive z-axes of either coordinate systems is oriented. Consult the work of A. J. Breen and B. Gault and team for further details.

**MONITOR**: (optional) *NXmonitor*

**atom_probe**: (required) *NXinstrument*

Metadata and numerical data of the atom probe and the lab in which it stands.

An atom probe microscope (experiment) is different compared to a large- scale facility or electron accelerator experiments in at least two ways:

- First, ionized atoms and molecular ion(s fragments) (in the case of atom probe tomography) and (primarily) imaging gas ions (in the case of field ion microscopy) are accelerated towards a position-sensitive and time-of-flight taking detector system. Hence, there is no real probe/beam.
- Second, the specimen is the lens of an atom probe microscope.

**instrument_name**: (required) *NX_CHAR*

Given name of the atom probe at the hosting institution. This is an alias. Examples could be LEAP5000, Raptor, Oxcart, one atom at a time, etc.

**location**: (optional) *NX_CHAR*

Location of the lab or place where the instrument is installed. Using GEOREF is preferred.

**flight_path_length**: (required) *NX_FLOAT* `{units=NX_LENGTH}`

The space inside the atom probe that ions pass through nominally when they leave the specimen and travel to the detector.

**field_of_view**: (optional) *NX_FLOAT* `{units=NX_LENGTH}`

**MANUFACTURER**: (required) *NXmanufacturer*

**analysis_chamber**: (optional) *NXchamber*

**load_lock_chamber**: (optional) *NXchamber*

**buffer_chamber**: (optional) *NXchamber*

**getter_pump**: (optional) *NXpump*

**roughening_pump**: (optional) *NXpump*

**turbomolecular_pump**: (optional) *NXpump*

**REFLECTRON**: (optional) *NXreflectron*

**applied**: (required) *NX_BOOLEAN*

Was the reflectron used?

**local_electrode**: (required) *NXlens_em*
A local electrode guiding the ion flight path.

**name**: (required) *NX_CHAR*

Identifier of the local_electrode in an e.g. database.

**APERTURE_EM**: (optional) *NXaperture_em*

**ion_detector**: (required) *NXdetector*

Detector for taking raw time-of-flight and ion/hit impact positions data.

**type**: (required) *NX_CHAR*

Description of the detector type. Specify if the detector is not the usual type, i.e. not a delay-line detector. In the case the detector is a multi-channel plate/ delay line detector, use mcp_dld. In the case the detector is a phosphor CCD use phosphor_ccd. In other case specify the detector type via free-text.

**name**: (recommended) *NX_CHAR*

Given name/alias.

**model**: (recommended) *NX_CHAR*

Given brand or model name by the manufacturer.

**serial_number**: (recommended) *NX_CHAR*

Given hardware name/serial number or hash identifier issued by the manufacturer.

**manufacturer_name**: (recommended) *NX_CHAR*

Given name of the manufacturer.

**signal_amplitude**: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_CURRENT*}

Amplitude of the signal detected on the multi-channel plate (MCP). This field should be used for storing the signal amplitude quantity within ATO files. The ATO file format is used primarily by the atom probe groups of the GPM in Rouen, France.

**pulser**: (required) *NXpulser_apm*

**pulse_mode**: (required) *NX_CHAR*

**pulse_frequency**: (required) *NX_NUMBER*

**pulse_fraction**: (required) *NX_NUMBER*

**pulsed_voltage**: (required) *NX_FLOAT*

**STAGE_LAB**: (required) *NXstage_lab*

**base_temperature**: (required) *NX_FLOAT* {units=*NX_TEMPERATURE*}

Average temperature at the specimen base, i.e. base temperature, during the measurement.

**control_software**: (optional) *NXcollection*

The majority of atom probe microscopes come from a single commercial manufacturer AMETEK (formerly Cameca). Their instruments are
controlled via an/a set of integrated instrument control system(s) (AP-Suite/IVAS/DAVis).

By contrast, instruments which were built by individual research groups such as of the French (GPM, Rouen, France), the Schmitz (Inspico, Stuttgart, Germany), the Feller (Oxcart, Erlangen, Germany), the Northwestern (D. Isheim, Seidman group et al.), or the PNNL group (Pacific Northwest National Laboratory, Portland, Oregon, U.S.) have other solutions to control the instrument. Some of which are modularized and open, some of which realize also integrated control units with portions of eventually undisclosed source code and (so far) lacking (support of)/open APIs.

Currently, there is no accepted/implemented community-specific API for getting finely granularized access to such control settings.

These considerations motivated the design of the NXapm application definition in that it stores quantities in NXcollection. groups to begin with. Holding heterogeneous, not yet standardized but relevant pieces of information is the purpose of this collection.

**analysis_chamber**: (optional) `NXcollection`

Track time-dependent settings over the course of the measurement about the environment in the analysis chamber such as gas pressure values etc.

**pressure**: (required) `NX_FLOAT {units=NX_PRESSURE}`

Average pressure in the analysis chamber.

**specimen_monitoring**: (optional) `NXcollection`

A place where details about the initial shape of the specimen can be stored. Ideally, here also data about the shape evolution of the specimen can be stored.

There are currently very few techniques which can measure the shape evolution:

- Correlative electron microscopy coupled with modeling but this usually takes an interrupted experiment in which the specimen is transferred, an image taken, and a new evaporation sequence initiated. Examples are I. Mouton et al. and C. Fletcher.
- Another, less accurate method, though, is to monitor the specimen evolution via the in-built camera system (if available) in the instrument.
- Another method is to use correlated scanning force microscopy methods like reported in C. Fleischmann.
- A continuous monitoring of the specimen in a correlative electron microscopy/atom probe experiment is planned to be developed by T. Kelly et al.

**initial_radius**: (required) `NX_FLOAT {units=NX_LENGTH}`

Ideally measured or best elaborated guess of the initial radius of the specimen.

**shank_angle**: (required) `NX_FLOAT {units=NX_ANGLE}`

Ideally measured or best elaborated guess of the shank angle. This is a measure of the specimen taper. Define it in such a way that the base of the specimen is modelled as a conical frustrum so that the shank angle
is the (shortest) angle between the specimen space z-axis and a vector on the lateral surface of the cone.

**ion_impact_positions**: (recommended) *NXprocess*

Details about where ions hit the ion_detector and data processing steps related to analog-to-digital conversion of detector signals into ion hit positions. For AMETEK LEAP instruments this processing takes place partly in the control unit of the detector partly in the software. The process is controlled by the acquisition/ instrument control software (IVAS/APSuite/DAVis). The exact details are not documented by AMETEK in an open manner. For instruments built by individual research groups, like the Oxcart instrument, individual timing data from the delay-line detector are openly accessible.

**program**: (required) *NX_CHAR*

Given name of the program that was used to perform this computation. Apart from the classical approach to use AMETEK software for this processing step, a number of open-source alternative tools exists.

**@version**: (required) *NX_CHAR*

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

**arrival_time_pairs**: (recommended) *NX_NUMBER* (Rank: 3, Dimensions: [n_ions, n_dld_wires, 2]) {units=*NX_TIME*}

Raw readings from the analog-to-digital-converter timing circuits of the detector wires.

**hit_positions**: (required) *NX_NUMBER* (Rank: 2, Dimensions: [n_ions, 2]) {units=*NX_LENGTH*}

Evaluated ion impact coordinates at the detector (either as computed from the arrival time data or as reported by the control software).

**detection_rate**: (required) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Average detection rate over the course of the experiment.

**hit_multiplicity**: (recommended) *NXprocess*

Data post-processing step which is, like the impact position analyses, also usually executed in the integrated control software. This processing yields how many ions were detected with each pulse.

It is possible that multiple ions evaporate and hit the same or different pixels of the detector on the same pulse. These data form the basis to analyses of the so-called (hit) multiplicity of an ion. Multiplicity must not be confused with how many atoms of the same element or isotope, respectively, a molecular ion contains (which is encoded with the isotope_vector field of each NXion instance.

**program**: (required) *NX_CHAR*

Given name of the program that was used to perform this computation. Apart from the classical approach to use AMETEK software for this processing step, a number of open-source alternative tools exists.
@version: (required) **NX_CHAR**

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

**pulses_since_last_ion**: (recommended) **NX_UINT** (Rank: 1, Dimensions: [n_ions]) {units=NX_UNITLESS}

Number of pulses since the last detected ion pulse. For multi-hit records, after the first record, this is zero.

**hit_multiplicity**: (recommended) **NX_UINT** (Rank: 1, Dimensions: [n_ions]) {units=NX_UNITLESS}

Hit multiplicity.

**pulse_id**: (optional) **NX_UINT** (Rank: 1, Dimensions: [n_ions]) {units=NX_UNITLESS}

Number of pulses since the start of the atom probe run/evaporation sequence.

**ion_filtering**: (recommended) **NXprocess**

Like impact position and hit multiplicity computations, ion filtering is a data post-processing step with which users identify which of the detected ions should be included in the voltage-and-bowl correction. This post-processing is usually performed via GUI interaction in the reconstruction pipeline of IVAS/APSuite.

**program**: (required) **NX_CHAR**

Given name of the program that was used to perform this computation. Apart from the classical approach to use AMETEK software for this processing step, a number of open-source alternative tools exists.

@version: (required) **NX_CHAR**

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

**evaporation_id_included**: (required) **NX_BOOLEAN** (Rank: 1, Dimensions: [n_ions])

Bitmask which is set to true if the ion is considered and false otherwise.

**voltage_and_bowl_correction**: (recommended) **NXprocess**

Data post-processing step to correct for ion impact position flight path differences, detector biases, and nonlinearities. This step is usually performed with commercial software.

**program**: (required) **NX_CHAR**

Given name of the program that was used to perform this computation. Apart from the classical approach to use AMETEK software for this processing step, a number of open-source alternative tools exists.
@version: (required) NX_CHAR

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

raw_tof: (recommended) NX_FLOAT (Rank: 1, Dimensions: [n_ions]) {units=NX_TIME}

Raw time-of-flight data as read out from the acquisition software if these data are available and accessible.

calibrated_tof: (required) NX_FLOAT (Rank: 1, Dimensions: [n_ions]) {units=NX_TIME}

Calibrated time-of-flight.

tof_calibration: (recommended) NXcollection

The key idea and algorithm of the voltage-and-bowl correction is qualitatively similar for instruments of different manufacturers or research groups.

Specific differences exists though in the form of different calibration models. For now we do not wish to resolve or generalize these differences. Rather the purpose of this collection is to provide a container where model-specific parameters and calibration models can be stored if users know these for sure.

For AMETEK LEAP atom probes this should be the place for storing initial calibration values. These values are accessible normally only by AMETEK service engineers. They use these for calibrating the detector and instrument.

Users can also use this NXcollection for storing the iteratively identified calibrations which scientists will get displayed in e.g. AP Suite while they execute the voltage-and-bowl correction as a part of the reconstruction pipeline in APSuite.

mass_to_charge_conversion: (recommended) NXprocess

Data post-processing step in which calibrated time-of-flight data (ToF) are interpreted into mass-to-charge-state ratios.

program: (required) NX_CHAR

Given name of the program that was used to perform this computation. Apart from the classical approach to use AMETEK software for this processing step, a number of open-source alternative tools exists.

@version: (required) NX_CHAR

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

mass_to_charge: (required) NX_FLOAT (Rank: 1, Dimensions: [n_ions]) {units=NX_ANY}
Mass-to-charge-state ratios

**parameter**: (recommended) *NXcollection*

Store vendor-specific calibration models here (if available).

**reconstruction**: (recommended) *NXprocess*

Data post-processing step to create a tomographic reconstruction of the specimen based on selected calibrated ion hit positions, the evaporation sequence, and voltage curve data. Very often scientists use own software scripts according to published procedures, so-called reconstruction protocols, i.e. numerical recipes how to compute x, y, z atomic positions from the input data.

**program**: (required) *NX_CHAR*

Given name of the program that was used to perform this computation. Similar comments as voltage_and_bowl_correction apply.

**@version**: (required) *NX_CHAR*

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

**protocol_name**: (recommended) *NX_CHAR*

Qualitative statement about which reconstruction protocol was used.

Any of these values:

- bas_original
- bas_modified
- geiser
- gault
- ivas
- apsuite
- other

**reconstructed_positions**: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_ions, 3]) \{units=*NX_LENGTH*\}

Three-dimensional reconstructed positions of the ions. Interleaved array of x, y, z positions in the specimen space.

**parameter**: (recommended) *NXcollection*

Different reconstruction protocols exist. Although these approaches are qualitatively similar, each protocol uses different parameters (and interprets these differently). The source code to IVAS/APSuite is not open. For now users should store reconstruction parameter in a collection.

**naive_point_cloud_density_map**: (required) *NXprocess*

To get a first overview of the reconstructed dataset, the format conversion computes a simple 3d histogram of the ion density using one nanometer cubic bins without applying smoothening algorithms on this histogram.
program: (required)  

Given name of the program that was used to perform this computation. Apart from the classical approach to use AMETEK software for this processing step, a number of open-source alternative tools exists.

@version: (required)  

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

DATA: (required)  

A default three-dimensional histogram of the total number of ions in each bin.

@long_name: (required)  

Naive point cloud density map.

counts: (required) N X_NUMBER  

(Rank: 3, Dimensions: [n_z, n_y, n_x]) {units=N X_UNITLESS}  

Array of counts for each bin.

zpos: (required) N X_NUMBER  

{units=N X_LENGTH}  

Bin positions along the z axis.

ypos: (required) N X_NUMBER  

{units=N X_LENGTH}  

Bin positions along the y axis.

xpos: (required) N X_NUMBER  

{units=N X_LENGTH}  

Bin positions along the x axis.

ranging: (recommended)  

Data post-processing step in which elemental, isotopic, and/or molecular identities are assigned to the ions. The documentation of these steps is based on ideas described in the literature:

- M. K. Miller
- D. Haley et al.
- M. Kühhbach et al.

program: (required)  

Given name of the program that was used to perform this computation. Apart from the classical approach to use AMETEK software for this processing step, a number of open-source alternative tools exists.

@version: (required)  

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.
**number_of_ion_types**: (required) \textit{NX_POSINT} \{units=\textit{NX_UNITLESS}\}

How many ion types are distinguished. If no ranging was performed each ion is of the special unknown type.

**maximum_number_of_atoms_per_molecular_ion**: (required) \textit{NX_UINT} \{units=\textit{NX_UNITLESS}\}

Assumed maximum value that suffices to store all relevant molecular ions, even the most complicated ones. Currently a value of 32 is used.

**mass_to_charge_distribution**: (recommended) \textit{NXprocess}

Specifies the computation of the mass-to-charge histogram. Usually mass-to-charge values are studied as an ensemble quantity, specifically these values are binned. The (\textit{NXprocess}) stores the settings of this binning.

**program**: (required) \textit{NX_CHAR}

Given name of the program that was used to perform this computation. Apart from the classical approach to use AMETEK software for this processing step, a number of open-source alternative tools exists.

@\textit{version}: (required) \textit{NX_CHAR}

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

**range_minmax**: (required) \textit{NX_FLOAT} \{Rank: 1, Dimensions: [2]\} \{units=\textit{NX_ANY}\}

Smallest and largest mass-to-charge value.

**range_increment**: (required) \textit{NX_FLOAT} \{units=\textit{NX_ANY}\}

Binning width

**mass_spectrum**: (required) \textit{NXdata}

A default histogram aka mass spectrum of the mass-to-charge-state ratio values.

@\textit{long_name}: (required) \textit{NX_CHAR}

Mass-to-charge-state histogram.

**counts**: (required) \textit{NX_NUMBER} \{Rank: 1, Dimensions: \text{[n_bins]}\} \{units=\textit{NX_UNITLESS}\}

Array of counts for each bin.

**bin_ends**: (required) \textit{NX_NUMBER} \{units=\textit{NX_ANY}\}

End of mass-to-charge-state ratio bin.

**background_quantification**: (recommended) \textit{NXprocess}

Details of the background model which was used to correct the total counts per bin into counts.

**program**: (required) \textit{NX_CHAR}
Given name of the program that was used to perform this computation. Apart from the classical approach to use AMETEK software for this processing step, a number of open-source alternative tools exists.

@version: (required) NX_CHAR

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

peak_search_and_deconvolution: (recommended) NXprocess

How where peaks in the background-corrected mass-to-charge histogram identified?

program: (required) NX_CHAR

Given name of the program that was used to perform this computation. Apart from the classical approach to use AMETEK software for this processing step, a number of open-source alternative tools exists.

@version: (required) NX_CHAR

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

PEAK: (required) NXpeak

peak_identification: (recommended) NXprocess

Details about how peaks, with taking into account error models, were interpreted as ion types or not.

program: (required) NX_CHAR

Given name of the program that was used to perform this computation. Apart from the classical approach to use AMETEK software for this processing step, a number of open-source alternative tools exists.

@version: (required) NX_CHAR

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

ION: (required) NXion

isotope_vector: (required) NX_UINT

charge_state: (required) NX_UINT

mass_to_charge_range: (required) NX_FLOAT (Rank: 2, Dimensions: [n_ranges, 2])
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXapm/ENTRY-
- /NXapm/ENTRY/atom_probe-
- /NXapm/ENTRY/atom_probe/analysis_chamber-
- /NXapm/ENTRY/atom_probe/buffer_chamber-
- /NXapm/ENTRY/atom_probe/control_software-
- /NXapm/ENTRY/atom_probe/control_software/analysis_chamber-
- /NXapm/ENTRY/atom_probe/field_of_view-
- /NXapm/ENTRY/atom_probe/flight_path_length-
- /NXapm/ENTRY/atom_probe/getter_pump-
- /NXapm/ENTRY/atom_probe/hit_multiplicity-
- /NXapm/ENTRY/atom_probe/hit_multiplicity/hit_multiplicity-
- /NXapm/ENTRY/atom_probe/hit_multiplicity/program-
- /NXapm/ENTRY/atom_probe/hit_multiplicity/program@version-
- /NXapm/ENTRY/atom_probe/hit_multiplicity/pulse_id-
- /NXapm/ENTRY/atom_probe/hit_multiplicity/pulses_since_last_ion-
- /NXapm/ENTRY/atom_probe/instrument_name-
- /NXapm/ENTRY/atom_probe/ion_detector-
- /NXapm/ENTRY/atom_probe/ion_detector/manufacturer_name-
- /NXapm/ENTRY/atom_probe/ion_detector/model-
- /NXapm/ENTRY/atom_probe/ion_detector/name-
- /NXapm/ENTRY/atom_probe/ion_detector/serial_number-
- /NXapm/ENTRY/atom_probe/ion_detector/signal_amplitude-
- /NXapm/ENTRY/atom_probe/ion_detector/type-
- /NXapm/ENTRY/atom_probe/ion_filtering-
- /NXapm/ENTRY/atom_probe/ion_filtering/evaporation_id_included-
- /NXapm/ENTRY/atom_probe/ion_filtering/program-
- /NXapm/ENTRY/atom_probe/ion_filtering/program@version-
- /NXapm/ENTRY/atom_probe/ion_impact_positions-
- /NXapm/ENTRY/atom_probe/ion_impact_positions/arrival_time_pairs-
- /NXapm/ENTRY/atom_probe/ion_impact_positions/detection_rate-
- /NXapm/ENTRY/atom_probe/ion_impact_positions/hit_positions-
- /NXapm/ENTRY/atom_probe/ion_impact_positions/program-
- /NXapm/ENTRY/atom_probe/ion_impact_positions/program@version-
• /NXapm/ENTRY/atom_probe/load_lock_chamber-group
• /NXapm/ENTRY/atom Probe/local_electrode-group
• /NXapm/ENTRY/atom Probe/local_electrode/APERTURE_EM-group
• /NXapm/ENTRY/atom Probe/local_electrode/name-field
• /NXapm/ENTRY/atom Probe/location-field
• /NXapm/ENTRY/atom Probe/MANUFACTURER-group
• /NXapm/ENTRY/atom Probe/mass_to_charge_conversion-group
• /NXapm/ENTRY/atom Probe/mass_to_charge_conversion/mass_to_charge-field
• /NXapm/ENTRY/atom Probe/mass_to_charge_conversion/parameter-group
• /NXapm/ENTRY/atom Probe/mass_to_charge_conversion/program-field
• /NXapm/ENTRY/atom Probe/mass_to_charge_conversion/program@version-attribute
• /NXapm/ENTRY/atom Probe/pulser-group
• /NXapm/ENTRY/atom Probe/pulser/pulse_fraction-field
• /NXapm/ENTRY/atom Probe/pulser/pulse_frequency-field
• /NXapm/ENTRY/atom Probe/pulser/pulse_mode-field
• /NXapm/ENTRY/atom Probe/pulser/pulsed_voltage-field
• /NXapm/ENTRY/atom Probe/ranging-group
• /NXapm/ENTRY/atom Probe/ranging/background_quantification-group
• /NXapm/ENTRY/atom Probe/ranging/background_quantification/program-field
• /NXapm/ENTRY/atom Probe/ranging/background_quantification/program@version-attribute
• /NXapm/ENTRY/atom Probe/ranging/mass_to_charge_distribution-group
• /NXapm/ENTRY/atom Probe/ranging/mass_to_charge_distribution/mass Spectrum-group
• /NXapm/ENTRY/atom Probe/ranging/mass_to_charge_distribution/mass_spectrum/bin_ends-field
• /NXapm/ENTRY/atom Probe/ranging/mass_to_charge_distribution/mass_spectrum/counts-field
• /NXapm/ENTRY/atom Probe/ranging/mass_to_charge_distribution/mass_spectrum@long_name-attribute
• /NXapm/ENTRY/atom Probe/ranging/mass_to_charge_distribution/program-field
• /NXapm/ENTRY/atom Probe/ranging/mass_to_charge_distribution/program@version-attribute
• /NXapm/ENTRY/atom Probe/ranging/mass_to_charge_distribution/range_increment-field
• /NXapm/ENTRY/atom Probe/ranging/mass_to_charge_distribution/range_minmax-field
• /NXapm/ENTRY/atom Probe/ranging/maximum_number_of_atoms_per_molecular_ion-field
• /NXapm/ENTRY/atom Probe/ranging/number_of_ion_types-field
• /NXapm/ENTRY/atom Probe/ranging/peak_identification-group
• /NXapm/ENTRY/atom Probe/ranging/peak_identification/ION-group
• /NXapm/ENTRY/atom Probe/ranging/peak_identification/ION/charge_state-field
• /NXapm/ENTRY/atom Probe/ranging/peak_identification/ION/isotope_vector-field
• /NXapm/ENTRY/atom Probe/ranging/peak_identification/ION/mass_to_charge_range-field
• /NXapm/ENTRY/atom_probe/ranging/peak_identification/program-field
• /NXapm/ENTRY/atom_probe/ranging/peak_identification/program@version-attribute
• /NXapm/ENTRY/atom_probe/ranging/peak_search_and_deconvolution-group
• /NXapm/ENTRY/atom_probe/ranging/peak_search_and_deconvolution/PEAK-group
• /NXapm/ENTRY/atom_probe/ranging/peak_search_and_deconvolution/program-field
• /NXapm/ENTRY/atom_probe/ranging/peak_search_and_deconvolution/program@version-attribute
• /NXapm/ENTRY/atom_probe/ranging/program-field
• /NXapm/ENTRY/atom_probe/ranging/program@version-attribute
• /NXapm/ENTRY/atom_probe/reconstruction-group
• /NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map-group
• /NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA-group
• /NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA-counts-field
• /NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA/xpos-field
• /NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA/ypos-field
• /NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA/zpos-field
• /NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA@long_name-attribute
• /NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/program-field
• /NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/program@version-attribute
• /NXapm/ENTRY/atom_probe/reconstruction/parameter-group
• /NXapm/ENTRY/atom_probe/reconstruction/program-field
• /NXapm/ENTRY/atom_probe/reconstruction/program@version-attribute
• /NXapm/ENTRY/atom_probe/reconstruction/protocol_name-field
• /NXapm/ENTRY/atom_probe/reconstruction/reconstructed_positions-field
• /NXapm/ENTRY/atom_probe/REFLECTRON-group
• /NXapm/ENTRY/atom_probe/REFLECTRON/applied-field
• /NXapm/ENTRY/atom_probe/roughening_pump-group
• /NXapm/ENTRY/atom_probe/specimen_monitoring-group
• /NXapm/ENTRY/atom_probe/specimen_monitoring/initial_radius-field
• /NXapm/ENTRY/atom_probe/specimen_monitoring/shank_angle-field
• /NXapm/ENTRY/atom_probe/STAGE_LAB-group
• /NXapm/ENTRY/atom_probe/STAGE_LAB/base_temperature-field
• /NXapm/ENTRY/atom_probe/turbomolecular_pump-group
• /NXapm/ENTRY/atom_probe/voltage_and_bowl_correction-group
• /NXapm/ENTRY/atom_probe/voltage_and_bowl_correction/calibrated_tof-field
• /NXapm/ENTRY/atom_probe/voltage_and_bowl_correction/program-field
• /NXapm/ENTRY/atom_probe/voltage_and_bowl_correction/program@version-attribute
- /NXapm/ENTRY/atom_probe/voltage_and_bowl_correction/raw_tof-field
- /NXapm/ENTRY/atom_probe/voltage_and_bowl_correction/tof_calibration-group
- /NXapm/ENTRY/COORDINATE_SYSTEM_SET-group
- /NXapm/ENTRY/DATA-group
- /NXapm/ENTRY/definition-field
- /NXapm/ENTRY/end_time-field
- /NXapm/ENTRY/experiment_description-field
- /NXapm/ENTRY/experiment_documentation-group
- /NXapm/ENTRY/experiment_identifier-field
- /NXapm/ENTRY/MONITOR-group
- /NXapm/ENTRY/operation_mode-field
- /NXapm/ENTRY/operator-group
- /NXapm/ENTRY/operator/address-field
- /NXapm/ENTRY/operator/affiliation-field
- /NXapm/ENTRY/operator/email-field
- /NXapm/ENTRY/operator/name-field
- /NXapm/ENTRY/operator/orcid-field
- /NXapm/ENTRY/operator/role-field
- /NXapm/ENTRY/operator/social_media_name-field
- /NXapm/ENTRY/operator/social_media_platform-field
- /NXapm/ENTRY/operator/telephone_number-field
- /NXapm/ENTRY/program-field
- /NXapm/ENTRY/program@version-attribute
- /NXapm/ENTRY/run_number-field
- /NXapm/ENTRY/specimen-group
- /NXapm/ENTRY/specimen/atom_types-field
- /NXapm/ENTRY/specimen/description-field
- /NXapm/ENTRY/specimen/name-field
- /NXapm/ENTRY/specimen/preparation_date-field
- /NXapm/ENTRY/specimen/sample_history-field
- /NXapm/ENTRY/specimen/short_title-field
- /NXapm/ENTRY/start_time-field
- /NXapm/ENTRY/thumbnail-group
- /NXapm/ENTRY/thumbnail@type-attribute
- /NXapm/ENTRY@version-attribute
NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm.nxdl.xml

**NXcalibration**

**Status:**
base class, extends NXobject

**Description:**
Subclass of NXprocess to describe post-processing calibrations.

**Symbols:**
The symbols used in the schema to specify e.g. dimensions of arrays

- `ncoeff`: Number of coefficients of the calibration function
- `nfeat`: Number of features used to fit the calibration function
- `ncal`: Number of points of the calibrated and uncalibrated axes

**Groups cited:**
none

**Structure:**

- `last_process`: (optional) `NX_CHAR`
  Indicates the name of the last operation applied in the NXprocess sequence.

- `applied`: (optional) `NX_BOOLEAN`
  Has the calibration been applied?

- `coefficients`: (optional) `NX_FLOAT` {units=`NX_ANY`}
  For non-linear energy calibrations, e.g. in a TOF, a polynomial function is fit to a set of features (peaks) at well defined energy positions to determine E(ToF). Here we can store the array of fit coefficients.

- `fit_function`: (optional) `NX_CHAR`
  For non-linear energy calibrations. Here we can store the formula of the fit function.
  Use a0, a1, ..., an for the coefficients, corresponding to the values in the coefficients field.
  Use x0, x1, ..., xn for the variables.
  The formula should be numpy compliant.

- `scaling`: (optional) `NX_FLOAT` {units=`NX_ANY`}
  For linear calibration. Scaling parameter.

- `offset`: (optional) `NX_FLOAT` {units=`NX_ANY`}
  For linear calibration. Offset parameter.

- `calibrated_axis`: (optional) `NX_FLOAT` {units=`NX_ANY`}
  A vector representing the axis after calibration, matching the data length

- `original_axis`: (optional) `NX_FLOAT` {units=`NX_ANY`}
  Vector containing the data coordinates in the original uncalibrated axis
description: (optional) NX_CHAR

A description of the procedures employed.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

• /NXcalibration/applied-field
• /NXcalibration/calibrated_axis-field
• /NXcalibration/coefficients-field
• /NXcalibration/description-field
• /NXcalibration/fit_function-field
• /NXcalibration/last_process-field
• /NXcalibration/offset-field
• /NXcalibration/original_axis-field
• /NXcalibration/scaling-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcalibration.nxdl.xml

NXchamber

Status:
base class, extends NXobject

Description:
Component of an instrument to store or place objects and specimens.

Symbols:
No symbol table

Groups cited:
NXmanufacturer

Structure:

name: (optional) NX_CHAR
Given name/alias.

description: (optional) NX_CHAR
Free-text field for describing details about the chamber. For example out of which material was the chamber built.

MANUFACTURER: (optional) Nxmanufacturer
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXchamber/description-field
- /NXchamber/MANUFACTURER-group
- /NXchamber/name-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXchamber.nxdl.xml

**NXcollectioncolumn**

**Status:**
- base class, extends NXobject

**Description:**
Subclass of NXelectronanalyser to describe the electron collection column of a photoelectron analyser.

**Symbols:**
- No symbol table

**Groups cited:**
NXaperture, NXdeflector, NXlens_em, NXtransformations

**Structure:**

- **scheme:** (optional) NX_CHAR
  
  Scheme of the electron collection lens, i.e. standard, deflector, PEEM, momentum microscope, etc.

- **extractor_voltage:** (optional) NX_FLOAT {units=NX_VOLTAGE}
  
  Voltage applied to the extractor lens

- **extractor_current:** (optional) NX_FLOAT {units=NX_CURRENT}
  
  Current necessary to keep the extractor lens at a set voltage. Variations indicate leakage, field emission or arc currents to the extractor lens.

- **working_distance:** (optional) NX_FLOAT {units=NX_LENGTH}
  
  Distance between sample and detector entrance

- **mode:** (optional) NX_CHAR
  
  Labelling of the lens setting in use.

- **projection:** (optional) NX_CHAR
  
  The space projected in the angularly dispersive directions, real or reciprocal
  
  Any of these values: real | reciprocal

- **magnification:** (optional) NX_FLOAT {units=NX_DIMENSIONLESS}
  
  The magnification of the electron lens assembly.

- **depends_on:** (optional) NX_CHAR
Specifies the position of the collection column by pointing to the last transformation in the transformation chain in the NXtransformations group.

**TRANSFORMATIONS**: (optional) *NXtransformations*

Collection of axis-based translations and rotations to describe the location and geometry of the deflector as a component in the instrument. Conventions from the NXtransformations base class are used. In principle, the McStas coordinate system is used. The first transformation has to point either to another component of the system or . (for pointing to the reference frame) to relate it relative to the experimental setup. Typically, the components of a system should all be related relative to each other and only one component should relate to the reference coordinate system.

**APERTURE**: (optional) *NXaperture*

The size and position of an aperture inserted in the column, e.g. field aperture or contrast aperture

**DEFLECTOR**: (optional) *NXdeflector*

Deflectors in the collection column section

**LENS_EM**: (optional) *NXlens_em*

Individual lenses in the collection column section

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXcollectioncolumn/APERTURE-group
- /NXcollectioncolumn/DEFLECTOR-group
- /NXcollectioncolumn/depends_on-field
- /NXcollectioncolumn/extractor_current-field
- /NXcollectioncolumn/extractor_voltage-field
- /NXcollectioncolumn/LENS_EM-group
- /NXcollectioncolumn/magnification-field
- /NXcollectioncolumn/mode-field
- /NXcollectioncolumn/projection-field
- /NXcollectioncolumn/scheme-field
- /NXcollectioncolumn/TRANSFORMATIONS-group
- /NXcollectioncolumn/working_distance-field

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcollectioncolumn.nxdl.xml
NXcontainer

**Status:**

base class, extends NXobject

**Description:**

State of a container holding the sample under investigation.

A container is any object in the beam path which absorbs the beam and whose contribution to the overall attenuation/scattering needs to be determined to process the experimental data. Examples of containers include glass capillary tubes, vanadium cans, windows in furnaces or diamonds in a Diamond Anvil Cell. The following figures show a complex example of a container:

![Diagram of a complex container setup](image)

**Fig. 12:** A hypothetical capillary furnace. The beam passes from left to right (blue dashes), passing through window 1, then window 2, before passing through the downstream wall of the capillary. It is then scattered by the sample with scattered beams passing through the upstream wall of the capillary, then windows 4 and 5. As part of the corrections for a PDF experiment it is necessary to subtract the PDF of the empty container (i.e. each of the windows and the capillary). To calculate the PDF of the empty container it is necessary to have the measured scattering data and to know the nature (e.g. density, elemental composition, etc.) of the portion of the container which the beam passed through.

![Diagram of container shapes](image)

**Fig. 13:** A complete description of the shapes of the container elements with their orientation relative to the beam and also information on whether they are upstream or downstream of the sample is also therefore important. For example, although the windows 2 and 4 have the same shape, the path taken through them by the beam is very different and this needs to be modelled. Furthermore, it is not inconceivable that windows might move during an experiment and thus the changes to the beampath would need to be accounted for.

This class encodes the position of the container with respect to the sample and allows the calculation of the beampath through the container. It also includes sufficient data to model beam absorption of the
container and a link to a dataset containing a measurement of the container with nothing inside, to allow
data corrections (at a specific beam energy/measurement time) to be made.

Symbols:
  No symbol table

Groups cited:
  NXbeam, NXshape, NXtransformations

Structure:
  name: (optional) NX_CHAR
  Descriptive name of container.
  description: (optional) NX_CHAR
  Verbose description of container and how it fits into the wider experimental set up.
  chemical_formula: (optional) NX_CHAR
  Chemical composition of the material the container is made from. Specified using CIF conven-
  tions. 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.
  density: (optional) NX_FLOAT (Rank: 1, Dimensions: [n_comp]) {units=NX_MASS_DENSITY}
  Density of the material the container is made from.
  packing_fraction: (optional) NX_FLOAT (Rank: 1, Dimensions: [n_comp]) {units=NX_UNITLESS}
  Fraction of the volume of the container occupied by the material forming the container.
  relative_molecular_mass: (optional) NX_FLOAT (Rank: 1, Dimensions: [n_comp]) {units=NX_MASS}
  Relative molecular mass of container.
  beam: (optional) NXbeam
  Details of beam incident on container, including the position relative to the sample (to determine
  whether the container is upstream or downstream of the sample).
  shape: (optional) NXshape
Shape of the container. In combination with orientation this should allow the beampath through the container to be modelled to allow the adsorption to be calculated.

**orientation**: (optional) *NXtransformations*

The angle the container makes to the beam and how it may change during the experiment. In combination with shape this should allow the beampath through the container to be modelled to allow the adsorption of the container to be calculated.

**reference_measurement**: *link* (suggested target: /NXentry)

A link to a full data collection which contains the actual measured data for this container within the experimental set up (with no sample or inner container(s)). This data set will also include the wavelength/energy, measurement time and intensity for which these data are valid.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXcontainer/beam-group
- /NXcontainer/chemical_formula-field
- /NXcontainer/density-field
- /NXcontainer/description-field
- /NXcontainer/name-field
- /NXcontainer/orientation-group
- /NXcontainer/packing_fraction-field
- /NXcontainer/reference_measurement-link
- /NXcontainer/relative_molecular_mass-field
- /NXcontainer/shape-group

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcontainer.nxdl.xml

**NXcoordinate_system_set**

**Status:**

base class, extends *NXobject*

**Description:**

Container to hold different coordinate systems conventions.

It is the purpose of this base class to define these conventions and offer a place to store mappings between different coordinate systems which are relevant for the interpretation of the data described by the application definition and base class instances.

For each Cartesian coordinate system users should use a set of NXtransformations:

- These should define the three base vectors.
- The location of the origin.
- The affine transformations which bring each axis of this coordinate system into registration with the McStas coordinate system.
• Equally, affine transformations should be given for the inverse mapping.

As an example one may take an experiment or computer simulation where there is a laboratory (lab) coordinate system, a sample/specimen coordinate system, a crystal coordinate system, and additional coordinate systems, which are eventually attached to components of the instrument.

If no additional transformation is specified in this group or if an instance of an NXcoordinate_system_set is absent it should be assumed the so-called McStas coordinate system is used.

Many application definitions in NeXus refer to this McStas coordinate system. This is a Cartesian coordinate system whose z axis points along the neutron propagation axis. The systems y axis is vertical up, while the x axis points left when looking along the z-axis. Thus, McStas is a right-handed coordinate system.

Within each NXtransformations a depends_on section is required. The depends_on field specifies if the coordinate system is the root/reference (which is indicated by writing “.” in the depends_on section.)

Symbols:

No symbol table

Groups cited:

NXtransformations

Structure:

TRANSFORMATIONS: (optional) NXtransformations

A group of transformations which specify:

• Three base vectors of the coordinate system.
• Origin of the coordinate system.
• A depends_on keyword. Its value can be “.” or the name of an NXtransformations instance which needs to exist in the NXcoordinate_system_set instance.
• If the coordinate system is the reference one it has to be named reference.

In case of having more than one NXtransformations there has to be for each additional coordinate system, i.e. the one not the reference:

• A set of translations and rotations which map each base vector to the reference.
• A set of translations and rotations which map each reference base vector to the coordinate system.

The NXtransformations for these mappings need to be formatted according to the descriptions in NXtransformations.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

• /NXcoordinate_system_set/TRANSFORMATIONS-group

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcoordinate_system_set.nxdl.xml
NXcorrector_cs

Status:
base class, extends NXobject

Description:
Corrector for aberrations in an electron microscope.
Different vendors use a different naming schemes for aberration coefficients. It is the users responsibility to map to the best matching values.
In transmission electron microscopes the spherical aberration corrector is a component that is controlled via instructing the microscope to achieve set point values. The corrector is composed of multiple lenses and other parts with vendor-specific details which are often undisclosed.
In the case of Nion Co. microscopes the coefficients of the corrector can be retrieved via NionSwift, which is why currently the Nion convention for the aberration coefficients is used.

Symbols:
No symbol table

Groups cited:
NXaberration, NXlens_em, NXmanufacturer, NXtransformations

Structure:

  applied: (optional) NX_BOOLEAN
  name: (optional) NX_CHAR
  description: (optional) NX_CHAR

  MANUFACTURER: (optional) NXmanufacturer
  ABERRATION: (optional) NXaberration
  LENS_EM: (optional) NXlens_em
  TRANSFORMATIONS: (optional) NXtransformations

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXcorrector_cs/ABERRATION-group
- /NXcorrector_cs/applied-field
- /NXcorrector_cs/description-field
- /NXcorrector_cs/LENS_EM-group
- /NXcorrector_cs/MANUFACTURER-group
- /NXcorrector_cs/name-field
**NXcsg**

**Status:**
- base class, extends *NXobject*

**Description:**
Constructive Solid Geometry base class, using *NXquadric* and *NXoff_geometry*

**Symbols:**
- No symbol table

**Groups cited:**
- *NXcsg*

**Structure:**

- **operation:** (optional) *NX_CHAR*
  - One of the standard construction solid geometry set operations, or if the CSG is a pointer to the geometry provided by an *NXquadric* or an *NXoff_geometry*. Takes values:
    - UNION
    - INTERSECTION
    - DIFFERENCE
    - COMPLEMENT
    - IS_QUADRIC
    - IS_MESH

- **geometry:** (optional) *NX_CHAR*
  - Path to a field that is either an *NXquadric* (if ‘operation’ = IS_QUADRIC) or an *NXoff_geometry* (if ‘operation’ = IS_MESH) that defines the surface making up the constructive solid geometry component. Compulsory if ‘operation’ is IS_QUADRIC or IS_MESH.

- **a:** (optional) *NXcsg*
  - The first operand of constructive solid geometry operation. Compulsory if ‘operation’ is UNION, INTERSECTION, DIFFERENCE or COMPLEMENT.

- **b:** (optional) *NXcsg*
  - The second operand of constructive solid geometry operation. Compulsory if ‘operation’ is UNION, INTERSECTION or DIFFERENCE.
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXcsg/a-group
- /NXcsg/b-group
- /NXcsg/geometry-field
- /NXcsg/operation-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcsg.nxdl.xml

NXcxi_ptycho

Status:
application definition, extends NXobject

Description:
Application definition for a ptychography experiment, compatible with CXI from version 1.6.

This is compatible with CXI from version 1.6 if this application definition is put at the top "entry" level. Above this a "cxi_version" field should be defined. The CXI format is name based, rather than class based, and so it is important to pay attention to the naming convention to be CXI compatible. There are duplications due to the format merger. These should be achieved by linking, with hdf5 Virtual Dataset being used to restructure any data that needs to be remapped. To be fully CXI compatible, all units (including energy) must be in SI units.

An example here is that CXI expects the data to always to have shape (npts_x*npts_y, frame_size_x, frame_size_y). For nexus this is only true for arbitrary scan paths with raster format scans taking shape (npts_x, npts_y, frame_size_x, frame_size_y).

Symbols:
These symbols will be used below to coordinate the shapes of the datasets.

npts_x: The number of points in the x direction
npts_y: Number of points in the y direction.
frame_size_x: Number of detector pixels in x
frame_size_y: Number of detector pixels in y

Groups cited:
NXbeam, NXcollection, NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample, NXsource, NXtransformations

Structure:
entry_1: (required) NXentry
    title: (optional) NX_CHAR
    start_time: (optional) NX_DATE_TIME
    end_time: (optional) NX_DATE_TIME
    definition: (required) NX_CHAR
Official NeXus NXDL schema to which this file conforms

Obligatory value: \texttt{NXcxi_ptycho}

\textbf{instrument\_1:} (required) \texttt{NXinstrument}

\textbf{source\_1:} (required) \texttt{NXsource}

- \textbf{name:} (required) \texttt{NX_CHAR}
- \textbf{energy:} (required) \texttt{NX_FLOAT}

  This is the energy of the machine, not the beamline.

- \textbf{probe:} (required) \texttt{NX_FLOAT}
- \textbf{type:} (required) \texttt{NX_FLOAT}

\textbf{beam\_1:} (required) \texttt{NXbeam}

- \textbf{energy:} (required) \texttt{NX_FLOAT}
  - \textbf{@units:} (required) \texttt{NX_CHAR}
- \textbf{extent:} (optional) \texttt{NX_FLOAT}
  - \textbf{@units:} (required) \texttt{NX_CHAR}

- \textbf{incident\_beam\_divergence:} (optional) \texttt{NX_FLOAT}
  - \textbf{@units:} (required) \texttt{NX_CHAR}

- \textbf{incident\_beam\_energy:} (required) \texttt{NX_FLOAT}
  - \textbf{@units:} (required) \texttt{NX_CHAR}

- \textbf{incident\_energy\_spread:} (required) \texttt{NX_FLOAT}
  - \textbf{@units:} (required) \texttt{NX_CHAR}

\textbf{detector\_1:} (required) \texttt{NXdetector}

- \textbf{@axes:} (required) \texttt{NX_CHAR}
  - should have value “[, data]”
- \textbf{@signal:} (required) \texttt{NX_CHAR}
  - should have value “data”

\textbf{translation:} (required) \texttt{NX_FLOAT} \{\texttt{units=NX_LENGTH}\}

  This is an array of shape (npts\_x*npts\_y, 3) and can be a Virtual Dataset of x and y

  - \textbf{@units:} (required) \texttt{NX_CHAR}
- \textbf{@axes:} (required) \texttt{NX_CHAR}

  this should take the value “translation:$slowaxisname:$fastaxisname”

- \textbf{@interpretation:} (required) \texttt{NX_CHAR}

  This should be “image”

\textbf{data:} (required) \texttt{NX_INT} (Rank: 3 for arbitrary scan, 4 for raster, Dimensions: [npts\_x, npts\_y, frame\_size\_x, frame\_size\_y])

\textbf{x\_pixel\_size:} (required) \texttt{NX_FLOAT} \{\texttt{units=NX_LENGTH}\}
@units: (required) NX_CHAR

y_pixel_size: (required) NX_FLOAT {units=NX_LENGTH}

@units: (required) NX_CHAR
distance: (required) NX_FLOAT {units=NX_LENGTH}

The distance between the detector and the sample

@units: (required) NX_CHAR
beam_center_x: (optional) NX_FLOAT {units=NX_LENGTH}

@units: (required) NX_CHAR
beam_center_y: (optional) NX_FLOAT {units=NX_LENGTH}

transformations: (required) NXtransformations

transformations: (required) NXtransformations

transformations: (required) NXtransformations

vector: (required) NX_NUMBER
data_1: link (suggested target: /NXentry/NXinstrument/NXdetector/data)

This data must always have shape (npts_x*npts_y, frame_size_x, frame_size_y) regardless of the scan pattern. Use hdf5 virtual dataset to achieve this.

MONITOR: (optional) NXmonitor

data: (required) NX_FLOAT (Rank: 1 for arbitrary scan, 2 for raster, Dimensions: [npts_x, npts_y])

DATA: (required) NXdata

@axes: (required) NX_CHAR

This should be “[x..]” for arbitrary scanning patterns, and “[x,...]” for raster

@signal: (required) NX_CHAR

This should be “data”
x_indices: (required) NX_CHAR

y_indices: (required) NX_CHAR

data: link (suggested target: /NXentry/NXinstrument/NXdetector/data)
x: link (suggested target: /NXentry/NXsample/NXtransformations/x)
y: link (suggested target: /NXentry/NXsample/NXtransformations/y)
data_1: (required) NXcollection

To ensure CXI compatibility the data in this group must always have shape that is (npts_x*npts_y, frame_size_x, frame_size_y). For nexus-style raster scans it is proposed that hdf5 virtual dataset is used.

data: link (suggested target: /NXentry/NXinstrument/NXdetector/data)
translation: link (suggested target: /NXentry/NXinstrument/NXdetector/translation)
sample_1: (required) NXsample
name: (optional) *NX_CHAR*

transformations: (required) *NXtransformations*

This must contain two fields with the x and y motors that are linked via the dependency
tree according to the real-life motor layout dependency. For raster scans x and y will
have shape (npts_x, npts_y) For arbitrary scans x and y will be (npts_x*npts_y,) An
attribute with the units for each motor is required.

@vector: (required) *NX_NUMBER*

geometry_1: (required) *NXcollection*

translation:  *link*  (suggested target:  /NXentry/NXinstrument/NXdetector/
translation)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXcxi_ptycho/DATA-group
- /NXcxi_ptycho/DATA/data-link
- /NXcxi_ptycho/DATA/x-link
- /NXcxi_ptycho/DATA/x_indices-field
- /NXcxi_ptycho/DATA/y-link
- /NXcxi_ptycho/DATA/y_indices-field
- /NXcxi_ptycho/DATA@axes-attribute
- /NXcxi_ptycho/DATA@signal-attribute
- /NXcxi_ptycho/data_1-group
- /NXcxi_ptycho/data_1/data-link
- /NXcxi_ptycho/data_1/translation-link
- /NXcxi_ptycho/entry_1-group
- /NXcxi_ptycho/entry_1/definition-field
- /NXcxi_ptycho/entry_1/end_time-field
- /NXcxi_ptycho/entry_1/instrument_1-group
- /NXcxi_ptycho/entry_1/instrument_1/beam_1-group
- /NXcxi_ptycho/entry_1/instrument_1/beam_1/energy-field
- /NXcxi_ptycho/entry_1/instrument_1/beam_1/energy@units-attribute
- /NXcxi_ptycho/entry_1/instrument_1/beam_1/extent-field
- /NXcxi_ptycho/entry_1/instrument_1/beam_1/extent@units-attribute
- /NXcxi_ptycho/entry_1/instrument_1/beam_1/incident_beam_divergence-field
- /NXcxi_ptycho/entry_1/instrument_1/beam_1/incident_beam_divergence@units-attribute
- /NXcxi_ptycho/entry_1/instrument_1/beam_1/incident_beam_energy-field
- /NXcxi_ptycho/entry_1/instrument_1/beam_1/incident_beam_energy@units-attribute
3.3. NeXus Class Definitions 471
NXdeflector

Status:

base class, extends NXobject

Description:

Deflectors as they are used e.g. in an electron analyser.

Symbols:

No symbol table

Groups cited:

NXtransformations

Structure:

**type**: (optional) **NX_CHAR**

Qualitative type of deflector with respect to the number of pole pieces

Any of these values: dipole | quadrupole | hexapole | octupole

**name**: (optional) **NX_CHAR**

Colloquial or short name for the deflector. For manufacturer names and identifiers use respective manufacturer fields.

**manufacturer_name**: (optional) **NX_CHAR**

Name of the manufacturer who built/constructed the deflector.

**manufacturer_model**: (optional) **NX_CHAR**

Hardware name, hash identifier, or serial number that was given by the manufacturer to identify the deflector.

**description**: (optional) **NX_CHAR**

Ideally an identifier, persistent link, or free text which gives further details about the deflector.

**voltage**: (optional) **NX_NUMBER** {units=NX_VOLTAGE}

Excitation voltage of the deflector. For dipoles it is a single number. For higher orders, it is an array.

**current**: (optional) **NX_NUMBER** {units=NX_CURRENT}

Excitation current of the deflector. For dipoles it is a single number. For higher orders, it is an array.

**depends_on**: (optional) **NX_CHAR**

Specifies the position of the deflector by pointing to the last transformation in the transformation chain in the NXtransformations group.

**TRANSFORMATIONS**: (optional) **NXtransformations**
Collection of axis-based translations and rotations to describe the location and geometry of the deflector as a component in the instrument. Conventions from the NXtransformations base class are used. In principle, the McStas coordinate system is used. The first transformation has to point either to another component of the system or (for pointing to the reference frame) to relate it relative to the experimental setup. Typically, the components of a system should all be related relative to each other and only one component should relate to the reference coordinate system.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXdeflector/current-field
- /NXdeflector/depends_on-field
- /NXdeflector/description-field
- /NXdeflector/manufacturer_model-field
- /NXdeflector/manufacturer_name-field
- /NXdeflector/name-field
- /NXdeflector/TRANSFORMATIONS-group
- /NXdeflector/type-field
- /NXdeflector/voltage-field

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXdeflector.nxdl.xml

**NXdistortion**

**Status:**

base class, extends NXobject

**Description:**

Subclass of NXprocess to describe post-processing distortion correction.

**Symbols:**

The symbols used in the schema to specify e.g. dimensions of arrays

- **nsym**: Number of symmetry points used for distortion correction
- **ndx**: Number of points of the matrix distortion field (x direction)
- **ndy**: Number of points of the matrix distortion field (y direction)

**Groups cited:**

none

**Structure:**

- **last_process**: (optional) NX_CHAR
  Indicates the name of the last operation applied in the NXprocess sequence.
- **applied**: (optional) NX_BOOLEAN
Has the distortion correction been applied?

**symmetry**: (optional) *NX_INT* {units=*NX_UNITLESS*}

For symmetry-guided distortion correction, where a pattern of features is mapped to the regular geometric structure expected from the symmetry. Here we record the number of elementary symmetry operations.

**original_centre**: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [2]) {units=*NX_UNITLESS*}

For symmetry-guided distortion correction. Here we record the coordinates of the symmetry centre point.

**original_points**: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [nsym, 2]) {units=*NX_UNITLESS*}

For symmetry-guided distortion correction. Here we record the coordinates of the relevant symmetry points.

**cdeform_field**: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [ndx, ndy]) {units=*NX_UNITLESS*}

Column deformation field for general non-rigid distortion corrections. 2D matrix holding the column information of the mapping of each original coordinate.

**rdeform_field**: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [ndx, ndy]) {units=*NX_UNITLESS*}

Row deformation field for general non-rigid distortion corrections. 2D matrix holding the row information of the mapping of each original coordinate.

**description**: (optional) *NX_CHAR*

Description of the procedures employed.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXdistortion/applied-field
- /NXdistortion/cdeform_field-field
- /NXdistortion/description-field
- /NXdistortion/last_process-field
- /NXdistortion/original_centre-field
- /NXdistortion/original_points-field
- /NXdistortion/rdeform_field-field
- /NXdistortion/symmetry-field

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXdistortion.nxdl.xml
NXebeam_column

Status:
base class, extends NXobject

Description:
Container for components to form a controlled electron beam.

Symbols:
No symbol table

Groups cited:
NXaperture_em, NXbeam, NXcorrector_cs, NXlens_em, NXmanufacturer, NXsensor, NXsource, NXstage_lab, NXtransformations

Structure:
MANUFACTURER: (optional) NXmanufacturer
electron_gun: (optional) NXsource
  The source which creates the electron beam.
  name: (optional) NX_CHAR
    Given name/alias.
voltage: (optional) NX_FLOAT {units=NX_VOLTAGE}
  Voltage relevant to compute the energy of the electrons immediately after they left the gun.
probe: (optional) NX_CHAR
  Type of radiation.
  Obligatory value: electron
emitter_type: (optional) NX_CHAR
  Emitter type used to create the beam.
  If the emitter type is other, give further details in the description field.
  Any of these values:
    • filament
    • schottky
    • cold_cathode_field_emitter
    • other
emitter_material: (optional) NX_CHAR
  Material of which the emitter is build, e.g. the filament material.
description: (optional) NX_CHAR
  Ideally, a (globally) unique persistent identifier, link, or text to a resource which gives further details.
MANUFACTURER: (optional) NXmanufacturer
TRANSFORMATIONS: (optional) NXtransformations
Affine transformation which detail the arrangement in the microscope relative to the optical axis and beam path.

- **APERTURE_EM**: (optional) `NXaperture_em`
- **LENS_EM**: (optional) `NXlens_em`
- **CORRECTOR_CS**: (optional) `NXcorrector_cs`
- **STAGE_LAB**: (optional) `NXstage_lab`
- **SENSOR**: (optional) `NXsensor`
  
  A sensor used to monitor an external or internal condition.

- **BEAM**: (optional) `NXbeam`
  
  Individual characterization results for the position, shape, and characteristics of the electron beam.
  
  NXtransformations should be used to specify the location of the position at which the beam was probed.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXebeam_column/APERTURE_EM-group`
- `/NXebeam_column/BEAM-group`
- `/NXebeam_column/CORRECTOR_CS-group`
- `/NXebeam_column/electron_gun-group`
- `/NXebeam_column/electron_gun/description-field`
- `/NXebeam_column/electron_gun/emitter_material-field`
- `/NXebeam_column/electron_gun/emitter_type-field`
- `/NXebeam_column/electron_gun/MANUFACTURER-group`
- `/NXebeam_column/electron_gun/name-field`
- `/NXebeam_column/electron_gun/probe-field`
- `/NXebeam_column/electron_gun/TRANSFORMATIONS-group`
- `/NXebeam_column/electron_gun/voltage-field`
- `/NXebeam_column/LENS_EM-group`
- `/NXebeam_column/MANUFACTURER-group`
- `/NXebeam_column/SENSOR-group`
- `/NXebeam_column/STAGE_LAB-group`

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXebeam_column.nxdl.xml
NXelectronanalyser

Status:

base class, extends NXobject

Description:

Subclass of NXinstrument to describe a photoelectron analyser.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays

nfa: Number of fast axes (axes acquired simultaneously, without scanning a physical quantity)
nsa: Number of slow axes (axes acquired scanning a physical quantity)

Groups cited:

NXcollectioncolumn, NXdeflector, NXdetector, NXenergydispersion, NXlens_em, NXspindispersion, NXtransformations

Structure:

description: (optional) NX_CHAR

Free text description of the type of the detector

name: (optional) NX_CHAR

Name or model of the equipment

@short_name: (optional) NX_CHAR

Acronym or other shorthand name

energy_resolution: (optional) NX_FLOAT [units=NX_ENERGY]

Energy resolution of the electron analyser (FWHM of gaussian broadening)

momentum_resolution: (optional) NX_FLOAT [units=NX_WAVENUMBER]

Momentum resolution of the electron analyser (FWHM)

angular_resolution: (optional) NX_FLOAT [units=NX_ANGLE]

Angular resolution of the electron analyser (FWHM)

spatial_resolution: (optional) NX_FLOAT [units=NX_LENGTH]

Spatial resolution of the electron analyser (Airy disk radius)

fast_axes: (optional) NX_CHAR (Rank: 1, Dimensions: [nfa])

List of the axes that are acquired simultaneously by the detector. These refer only to the experimental variables recorded by the electron analyser. Other variables such as temperature, manipulator angles etc. are labeled as fast or slow in the data.

3.3. NeXus Class Definitions 477
Table 1: Examples

<table>
<thead>
<tr>
<th>Mode</th>
<th>fast_axes</th>
<th>slow_axes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hemispherical in ARPES mode</td>
<td>['energy', 'kx']</td>
<td>['energy']</td>
</tr>
<tr>
<td>Hemispherical with channeltron, sweeping energy mode</td>
<td></td>
<td>['energy']</td>
</tr>
<tr>
<td>Tof</td>
<td>['energy', 'kx', 'ky']</td>
<td></td>
</tr>
<tr>
<td>Momentum microscope, spin-resolved</td>
<td>['energy', 'kx', 'ky']</td>
<td>['spin up-down', 'spin left-right']</td>
</tr>
</tbody>
</table>

Axes may be less abstract than this, i.e. ['detector_x', 'detector_y']. If energy_scan_mode=sweep, fast_axes: ['energy', 'kx']; slow_axes: ['energy'] is allowed.

**slow_axes**: (optional) **NX_CHAR** (Rank: 1, Dimensions: [nsa])

List of the axes that are acquired by scanning a physical parameter, listed in order of decreasing speed. See fast_axes for examples.

**depends_on**: (optional) **NX_CHAR**

Refers to the last transformation specifying the position of the manipulator in the NXtransformations chain.

**TRANSFORMATIONS**: (optional) **NXtransformations**

Collection of axis-based translations and rotations to describe the location and geometry of the manipulator as a component in the instrument. Conventions from the NXtransformations base class are used. In principle, theStas coordinate system is used. The first transformation has to point either to another component of the system or . (for pointing to the reference frame) to relate it relative to the experimental setup. Typically, the components of a system should all be related relative to each other and only one component should relate to the reference coordinate system.

**COLLECTIONCOLUMN**: (optional) **NXcollectioncolumn**

Describes the electron collection (spatial and momentum imaging) column

**ENERGYDISPERSION**: (optional) **NXenergydispersion**

Describes the energy dispersion section

**SPINDISPERSION**: (optional) **NXspindispersion**

Describes the spin dispersion section

**DETECTOR**: (optional) **NXdetector**

Describes the electron detector

**DEFLECTOR**: (optional) **NXdeflector**

Deflectors outside the main optics ensambles described by the subclasses

**LENS_EM**: (optional) **NXlens_em**

Individual lenses outside the main optics ensambles described by the subclasses
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXelectronanalyser/angular_resolution-field
- /NXelectronanalyser/COLLECTIONCOLUMN-group
- /NXelectronanalyser/DEFLECTOR-group
- /NXelectronanalyser/depends_on-field
- /NXelectronanalyser/description-field
- /NXelectronanalyser/DETECTOR-group
- /NXelectronanalyser/energy_resolution-field
- /NXelectronanalyser/ENERGYDISPERSION-group
- /NXelectronanalyser/fast_axes-field
- /NXelectronanalyser/LENS_EM-group
- /NXelectronanalyser/momentum_resolution-field
- /NXelectronanalyser/name-field
- /NXelectronanalyser/name@short_name-attribute
- /NXelectronanalyser/slow_axes-field
- /NXelectronanalyser/spatial_resolution-field
- /NXelectronanalyser/SPINDISPERSSION-group
- /NXelectronanalyser/TRANSFORMATIONS-group

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXelectronanalyser.nxdl.xml

NXelectrostatic_kicker

Status:
base class, extends NXobject

Description:
definition for a electrostatic kicker.

Symbols:
No symbol table

Groups cited:
NXlog

Structure:

defscription: (optional) NX_CHAR
extended description of the kicker.

beamline_distance: (optional) NX_FLOAT {units=NX_LENGTH}
define position of beamline element relative to production target
timing: (optional) `NX_FLOAT {units=NX_TIME}`
   kicker timing as defined by description attribute
   @description: (optional) `NX_CHAR`

set_current: (optional) `NX_FLOAT {units=NX_CURRENT}`
   current set on supply.

set_voltage: (optional) `NX_FLOAT {units=NX_VOLTAGE}`
   voltage set on supply.

read_current: (optional) `NXlog`
   current read from supply.
   value: (optional) `NX_CHAR {units=NX_CURRENT}`

read_voltage: (optional) `NXlog`
   voltage read from supply.
   value: (optional) `NX_CHAR {units=NX_VOLTAGE}`

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXelectrostatic_kicker/beamline_distance-field
- /NXelectrostatic_kicker/description-field
- /NXelectrostatic_kicker/read_current-group
- /NXelectrostatic_kicker/read_current/value-field
- /NXelectrostatic_kicker/read_voltage-group
- /NXelectrostatic_kicker/read_voltage/value-field
- /NXelectrostatic_kicker/set_current-field
- /NXelectrostatic_kicker/set_voltage-field
- /NXelectrostatic_kicker/timing-field
- /NXelectrostatic_kicker/timing@description-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXelectrostatic_kicker.nxdl.xml

NXellipsometry

Status:
   application definition, extends `NXobject`

Description:
   Ellipsometry, complex systems, up to variable angle spectroscopy.
   Information on ellipsometry is provided, e.g. in:


Open access sources:

• https://www.angstromadvanced.com/resource.asp
• https://pypolar.readthedocs.io/en/latest/

Review articles:


Symbols:

Variables used throughout the document, e.g. dimensions and important parameters

N_wavelength: Size of the energy or wavelength vector used, the length of NXinstrument/spectrometer/wavelength array

N_variables: How many variables are saved in a measurement. e.g. 2 for Psi and Delta, 16 for Mueller matrix elements, 15 for normalized Mueller matrix, 3 for NCS, the length of NXsample/column_names

N_angles: Number of incident angles used, the length of NXinstrument/angle_of_incidence array

N_p1: Number of sample parameters scanned, if you varied any of the parameters such as temperature, pressure, or pH, the maximal length of the arrays specified by NXsample/environment_conditions/SENSOR/value if it exists.

N_time: Number of time points measured, the length of NXsample/time_points

Groups cited:

NXaperture, NXdata, NXdetector, NXentry, NXenvironment, NXgrating, NXinstrument, NXmonochromator, NXprocess, NXsample, NXsensor, NXslit, NXsource, NXsubentry, NXtransformations, NXuser

Structure:

ENTRY: (required) NXentry

This is the application definition describing ellipsometry experiments.

Such experiments may be as simple as identifying how a reflected beam of light with a single wavelength changes its polarization state, to a variable angle spectroscopic ellipsometry experiment.

The application definition defines:

• elements of the experimental instrument

• calibration information if available

3.3. NeXus Class Definitions

481
• parameters used to tune the state of the sample
• sample description

definition: (required) NX_CHAR
An application definition for ellipsometry.
Obligatory value: NXellipsometry
@version: (required) NX_CHAR
   Version number to identify which definition of this application definition was used for this entry/data.
@url: (required) NX_CHAR
   URL where to find further material (documentation, examples) relevant to the application definition

experiment_identifier: (required) NX_CHAR
Unique identifier of the experiment, such as a (globally persistent) unique identifier. i) The identifier is usually defined by the facility or principle investigator. ii) The identifier enables to link experiments to e.g. proposals.

experiment_description: (recommended) NX_CHAR
A free-text description of the experiment. What is the aim of the experiment? The general procedure.

start_time: (required) NX_DATE_TIME
Start time of the experiment. UTC offset should be specified.

acquisition_program: (optional) NXprocess
@url: (required) NX_CHAR
   Website of the software.
program: (required) NX_CHAR
   Commercial or otherwise defined given name to the program that was used to generate the result file(s) with measured data and metadata. This program converts the measured signals to ellipsometry data. If home written, one can provide the actual steps in the NOTE subfield here.
version: (required) NX_CHAR
   Either version with build number, commit hash, or description of a (online) repository where the source code of the program and build instructions can be found so that the program can be configured in such a way that result files can be created ideally in a deterministic manner.

operator: (required) NXuser
Contact information of at least the user of the instrument or the investigator who performed this experiment. Adding multiple users if relevant is recommended.

name: (required) NX_CHAR
   Name of the user.
affiliation: (required) NX_CHAR
Name of the affiliation of the user at the point in time when the experiment was performed.

**address**: (required) *NX_CHAR*

Full address (street, street number, ZIP, city, country) of the user’s affiliation.

**email**: (required) *NX_CHAR*

Email address of the user.

**orcid**: (recommended) *NX_CHAR*

Author ID defined by https://orcid.org/.

**telephone_number**: (recommended) *NX_CHAR*

Official telephone number of the user.

**INSTRUMENT**: (required) *NXinstrument*

General properties of the ellipsometry equipment

**model**: (required) *NX_CHAR*

The name of the instrument

**@version**: (required) *NX_CHAR*

The used version of the hardware if available. If not a commercial instrument use date of completion of the hardware.

**company**: (optional) *NX_CHAR*

Name of the company which build the instrument

**construction_year**: (optional) *NX_DATE_TIME*

ISO8601 date when the instrument was constructed. UTC offset should be specified.

**firmware**: (required) *NX_CHAR*

Commercial or otherwise defined name of the software that was used for the measurement

**@version**: (required) *NX_CHAR*

Version and build number or commit hash of the software source code

**@url**: (required) *NX_CHAR*

Website of the software.

**focussing_probes**: (required) *NX_BOOLEAN*

Were focussing probes (lenses) used?

**data_correction**: (optional) *NX_BOOLEAN*

Were the recorded data corrected by the window effects of the lenses?

**angular_spread**: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Specify the angular spread caused by the focussing probes

**ellipsometry_type**: (required) *NX_CHAR*
What type of ellipsometry was used? See Fujiwara Table 4.2
Any of these values:
• rotating analyzer
• rotating analyzer with analyzer compensator
• rotating analyzer with polarizer compensator
• rotating polarizer
• rotating compensator on polarizer side
• rotating compensator on analyzer side
• modulator on polarizer side
• modulator on analyzer side
• dual compensator
• phase modulation
• imaging ellipsometry
• null ellipsometry

calibration_status: (required) NX_CHAR
Was a calibration performed? If yes, when was it done? If the calibration time is provided, it should be specified in calibration/calibration_time.
Any of these values:
• calibration time provided
• no calibration
• within 1 hour
• within 1 day
• within 1 week

angle_of_incidence: (required) NX_NUMBER (Rank: 1, Dimensions: [N_angles])  
{units=NX_ANGLE}
Incident angle of the beam vs. the normal of the bottom reflective (substrate) surface in the sample

light_source: (required) NXsource
Specify the used light source. Multiple selection possible.

calibration: (recommended) NXsubentry
Ellipsometers require regular calibration to adjust the hardware parameters for proper zero values and background light compensation.

calibration_time: (optional) NX_DATE_TIME
If calibration status is ‘calibration time provided’, specify the ISO8601 date when calibration was last performed before this measurement. UTC offset should be specified.

calibration_sample: (required) NX_CHAR
Free-text to describe which sample was used for calibration, e.g. silicon wafer with 25 nm thermal oxide layer.

**calibration_data**: (required) *NXsubentry*

Arrays which provide the measured calibration data. Multiple sets are possible, e.g. Psi and delta measured on a e.g. silicon calibration wafer, and the straight-through data. We recommend to provide data that is measured under the same settings as the measurement was performed, that is if Psi and Delta are measured for your data, also provide Psi and Delta here and use the same wavelengths as for the measured data.

**calibration_data_type**: (required) *NX_CHAR*

What data were recorded for the calibration? The number of variables (N_variables) have to be set to the number of provided data columns accordingly, e.g. psi/delta -> N_variables = 2, Jones vector -> N_variables = 4, Mueller matrix -> N_variables = 16, etc.

Any of these values:

• psi/delta
• tan(psi)/cos(delta)
• Jones matrix
• Mueller matrix
• not provided

**calibration_angle_of_incidence**: (required) *NX_NUMBER* (Rank: 1, Dimensions: [N_calibration_angles]) {units=*NX_ANGLE*}

Angle(s) of incidence used during the calibration measurement (excluding straight through mode)

**calibration_wavelength**: (required) *NX_NUMBER* (Rank: 1, Dimensions: [N_calibration_wavelength])

The wavelength or equivalent values (which are inter-convertible). The importer should convert all to one unit, and make the others accessible. Historically, energy is used in eV, but for visible spectroscopy wavelength is more common, for IR wave numbers in 1/cm units.

Possibly use the same type of data as for the measurement.

**calibration_data**: (required) *NX_NUMBER* (Rank: 3, Dimensions: [N_calibration_angles+1, N_variables, N_calibration_wavelength]) {units=*NX_UNITLESS*}

Calibration is performed on a reference surface (usually a silicon wafer with a well defined oxide layer) at a number of angles of incidence and in a straight through mode (transmission in air).

**stage**: (required) *NXsubentry*

Sample stage, holding the sample at a specific position in X,Y,Z (Cartesian) coordinate system and at an orientation defined by three Euler angles (alpha, beta, gamma). The stage may be motorized or manual, special for liquids or gas environment.

**stage_type**: (required) *NX_CHAR*
Specify what type of stage was used.

Any of these values:

- manual stage
- scanning stage
- liquid stage
- gas cell
- cryostat

**description**: (recommended) *NX_CHAR*

A free-text field to provide information about the stage.

**TRANSFORMATIONS**: (recommended) *NXtransformations*

The stage coordinate system vs. the incident beam. The Z-axis of the stage is considered to point along the normal of the substrate (bottom reflecting surface) from the stage towards the general direction of the light source. The beam comes with the angle of incidence towards this Z-axis, but in opposite direction, thus they are connected with a rotation of 180 - angle of incidence (in degrees). This transformation brings us from the NEXUS coordinates to the stage coordinates. Then provide the set of translations (if there are any). These all have a vector defining their relative direction in the current coordinate system. (This current coordinate system changes with every transformation if you set the parameter ‘depends’ to the name of the previous step.) Last, provide the rotations of the sample.

**alternative**: (optional) *NX_CHAR*

If there is no motorized stage, we should at least qualify where the beam hits the sample and in what direction the sample stands in a free-text description, e.g. ‘center of sample, long edge parallel to plane of incidence’.

**window**: (optional) *NXaperture*

For environmental measurements, the environment (liquid, vapor, vacuum etc.) is enclosed in a cell or cryostat, which has windows both in the direction of the source and the detector (looking from the sample). These windows also add a phase shift to the light altering the measured signal. This shift has to be corrected based on measuring a known sample in the environmental cell.

**material**: (required) *NX_CHAR*

The material of the window

Any of these values:

- quartz
- diamond
- calcium fluoride
- zinc selenide
- thallium bromoiodide
- alkali halide compound
• Mylar
• other

**other_material**: (optional) *NX_CHAR*

If you specified ‘other’ as window material, describe here what it is.

**thickness**: (required) *NX_NUMBER* {units=*NX_LENGTH*}

Thickness of the window

**orientation_angle**: (required) *NX_NUMBER* {units=*NX_ANGLE*}

Angle of the window normal (outer) vs. the substrate normal (similar to the angle of incidence).

**reference_data**: (required) *NXsubentry*

Recorded data that can be used to calculate the window effect. Typically this is the substrate (e.g. silicon with thermal oxide layer) in air without window and in a known medium with the window.

**reference_sample**: (required) *NX_CHAR*

What sample was used to estimate the window effect?

**reference_wavelength**: (required) *NX_NUMBER* (Rank: 1, Dimensions: [N_wavelength]) {units=*NX_LENGTH*}

Wavelength of the reference data. Use the same wavelengths at which all other measurements are recorded

**data**: (recommended) *NX_NUMBER* (Rank: 4, Dimensions: [2, N_angles, N_variables, N_wavelength]) {units=*NX_UNITLESS*}

Recorded data of a reference surface with and without window/medium.

**DETECTOR**: (required) *NXdetector*

Which type of detector was used, and what is known about it? A detector can be a photomultiplier (PMT), a CCD in a camera, or an array in a spectrometer. If so, the whole detector unit goes in here. Integration time is the count time field, or the real time field. See their definition.

**detector_type**: (required) *NX_CHAR*

What kind of detector module is used, e.g. CCD-spectrometer, CCD camera, PMT, photodiode, etc.

Any of these values:

• PMT
• photodiode
• avalanche diode
• CCD camera
• CCD spectrometer
• other

**other_detector**: (optional) *NX_CHAR*

If you specified ‘other’ as detector type, please write down what it is.
**revolution**: (optional) `NX_NUMBER {units=NX_ANY}`

Define how many rotations of the rotating element were taken into account per spectrum.

**rotating_element**: (required) `NX_CHAR`

Define which element rotates, e.g. polarizer or analyzer.

Any of these values:
- polarizer (source side)
- analyzer (detector side)
- compensator (source side)
- compensator (detector side)

**fixed_revolution**: (optional) `NX_NUMBER {units=NX_FREQUENCY}`

Rotation rate, if the revolution does not change during the measurement.

**variable_revolution**: (optional) `NX_NUMBER` (Rank: 1, Dimensions: [2])

Specify maximum and minimum values for the revolution.

**intensity_threshold**: (optional) `NX_NUMBER {units=NX_UNITLESS}`

Minimum signal for which dynamic averaging is performed.

**min_intensity**: (optional) `NX_NUMBER {units=NX_UNITLESS}`

Value for the minimum intensity chosen. Data points below this value might be skipped by the instrument

**spectrometer**: (required) `NXmonochromator`

The spectroscope element of the ellipsometer before the detector, but often integrated to form one closed unit. Information on the dispersive element can be specified in the subfield GRATING. Note that different gratings might be used for different wavelength ranges. The dispersion of the grating for each wavelength range can be stored in grating_dispersion.

**wavelength**: (required) `NX_NUMBER` (Rank: 1, Dimensions: [N_wavelength]) `{units=NX_LENGTH}`

Wavelength value(s) used for the measurement. An array of 1 or more elements. Length defines N_wavelength

**spectral_resolution**: (optional) `NX_NUMBER` `{units=NX_WAVENUMBER}`

Spectral resolution of the instrument.

**GRATING**: (optional) `NXgrating`

Diffraction grating, as could be used in a monochromator. If two or more gratings were used, define the angular dispersion and the wavelength range (min/max wavelength) for each grating and make sure that the wavelength ranges do not overlap. The dispersion should be defined for the entire wavelength range of the experiment.

**angular Dispersion**: (optional) `NX_NUMBER`

Dispersion of the grating in nm/mm used.
grating_wavelength_min: (optional) NX_NUMBER
{units=NX_LENGTH}

Minimum wavelength of the grating.

grating_wavelength_max: (optional) NX_NUMBER
{units=NX_LENGTH}

Maximum wavelength of the grating.

SLIT: (optional) NXslit

Define the width of the monochromator slit in the subfield x_gap.

fixed_slit: (optional) NX_BOOLEAN

Was the slit width fixed?

max_gap: (optional) NX_NUMBER {units=NX_LENGTH}

If slit width was not fixed, define the maximum slit width.

SAMPLE: (required) NXsample

Properties of the sample, its history, the sample environment and experimental conditions (e.g. surrounding medium, temperature, pressure etc.), along with the data (data type, wavelength array, measured data).

atom_types: (required) NX_CHAR

Use Hill’s system for listing elements of the periodic table which are inside or attached to the surface of the specimen and thus relevant from a scientific point. The purpose of this field is to allow material databases to parse the relevant elements without having to interpret the sample history or other fields.

sample_name: (required) NX_CHAR

Descriptive name of the sample

sample_history: (required) NX_CHAR

Ideally, a reference to the location or a unique (globally persistent) identifier (e.g.) of e.g. another file which gives as many as possible details of the material, its microstructure, and its thermo-chemo-mechanical processing/preparation history. In the case that such a detailed history of the sample is not available, use this field as a free-text description to specify details of the sample and its preparation.

preparation_date: (recommended) NX_DATE_TIME

ISO8601 date with time zone (UTC offset) specified.

layer_structure: (required) NX_CHAR

Qualitative description of the layer structure for the sample. For example: Si/native oxide/thermal oxide/polymer/peptide

data_identifier: (required) NX_NUMBER

An identifier to correlate data to the experimental conditions, if several were used in this measurement; typically an index of 0 - N

data_type: (required) NX_CHAR
Select which type of data was recorded, for example Psi and Delta (see: https://en.wikipedia.org/wiki/Ellipsometry#Data_acquisition). It is possible to have multiple selections. Data types may also be converted to each other, e.g., a Mueller matrix contains N,C,S data as well. This selection defines how many columns (N_variables) are stored in the data array.

Any of these values:
- psi/delta
- \( \tan(\psi) / \cos(\delta) \)
- Mueller matrix
- Jones matrix
- N/C/S
- raw data

column_names: (required) \( \text{NX\_CHAR} \) (Rank: 1, Dimensions: \([\text{N\_variables}]\))

Please list in this array the column names used in your actual data. That is ['\psi', '\delta'] or ['MM1', 'MM2', 'MM3', ..., 'MM16'] for a full Mueller matrix, etc.

measured_data: (required) \( \text{NX\_NUMBER} \) (Rank: 5, Dimensions: \([\text{N\_time}, \text{N\_p1}, \text{N\_angles}, \text{N\_variables}, \text{N\_wavelength}]\))

Resulting data from the measurement, described by data type. Minimum two columns containing Psi and Delta, or for the normalized Mueller matrix it may be 16 (or 15 if the element (1,1) is all 1).

data_error: (recommended) \( \text{NX\_NUMBER} \) (Rank: 5, Dimensions: \([\text{N\_time}, \text{N\_p1}, \text{N\_angles}, \text{N\_variables}, \text{N\_wavelength}]\))

Specified uncertainties (errors) of the data described by data type. The structure is the same as for the measured data.

time_points: (optional) \( \text{NX\_NUMBER} \) (Rank: 1, Dimensions: \([\text{N\_time}]\)) \{units=\text{NX\_TIME}\}

An array of relative time points if a time series was recorded.

environment_conditions: (required) \( \text{NX\_environment} \)

Specify external parameters that have influenced the sample.

medium: (required) \( \text{NX\_CHAR} \)

Describe what was the medium above or around the sample. The common model is built up from the substrate to the medium on the other side. Both boundaries are assumed infinite in the model. Here, define the name of the medium (e.g., water, air, UHV, etc.).

medium_refractive_indices: (optional) \( \text{NX\_NUMBER} \) (Rank: 1, Dimensions: \([\text{N\_wavelength}]\)) \{units=\text{NX\_UNITLESS}\}

Array of pairs of complex refractive indices of the medium for every measured wavelength. Only necessary if the measurement was performed not in air, or something very well known, e.g., high purity water. Specify the complex refractive index: \( n + ik \)

number_of_runs: (optional) \( \text{NX\_UINT} \) \{units=\text{NX\_DIMENSIONLESS}\}
How many measurements were done varying the parameters? This forms an extra dimension beyond incident angle, time points and energy/wavelength (this is the length of the 4th dimension of the data). Defaults to 1.

**varied_parameters**: (optional) `NX_CHAR`

Indicates which parameter was changed. Its definition must exist below. The specified variable has to be number_of_runs long, providing the parameters for each data set. If you vary more than one parameter simultaneously use one signal instance for each. Record every parameter value in a linear manner, so N_p1 is the number of measurements taken. For example, if you measure at two temperatures and three pressures the temperature signal value looks like [T1, T1, T1, T2, T2, T2] and the pressure signal value looks like [p1, p2, p3, p1, p2, p3], and N_p1 = 6.

Any of these values:
- optical excitation
- voltage
- temperature
- pH
- stress
- stage positions

**optical_excitation**: (optional) `NXsource`

Was the sample modified using an optical source? Describe in this group the parameters of the optical excitation used.

**wavelength**: (required) `NX_NUMBER` `{units=NX_LENGTH}`

Wavelength value(s) or the range used for excitation. In cases of continuous laser radiation, a value or a set of values may do but for other illumination types, such as pulsed lasers, or lamps, a range may describe the source better.

**broadening**: (optional) `NX_NUMBER` `{units=NX_LENGTH}`

Specify the FWHM of the excitation

**duration**: (optional) `NX_NUMBER` `{units=NX_TIME}`

How long was the sample excited.

**pulse_energy**: (optional) `NX_NUMBER` `{units=NX_ENERGY}`

The integrated energy of light pulse.

**SENSOR**: (optional) `NXsensor`

A sensor used to monitor an external condition. The value field contains the measured values. If it is constant within an error for every run then use only an array of length one.

**derived_parameters**: (optional) `NXprocess`

What parameters are derived from the above data.

**depolarization**: (optional) `NX_NUMBER` `{units=NX_UNITLESS}`
Light loss due to depolarization as a value in [0-1].

**plot**: (optional) NXdata

A default view of the data, in this case Psi vs. wavelength and the angles of incidence. If Psi does not exist, use other Mueller matrix elements, such as N, C and S.

**@axes**: (required) NX_CHAR

We recommend to use wavelength as a default attribute, but it can be replaced in the case of not full spectral ellipsometry to any suitable parameter along the X-axis.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXellipsometry/ENTRY-group
- /NXellipsometry/ENTRY/acquisition_program-group
- /NXellipsometry/ENTRY/acquisition_program/program-field
- /NXellipsometry/ENTRY/acquisition_program/version-field
- /NXellipsometry/ENTRY/acquisition_program@url-attribute
- /NXellipsometry/ENTRY/definition-field
- /NXellipsometry/ENTRY/definition@url-attribute
- /NXellipsometry/ENTRY/definition@version-attribute
- /NXellipsometry/ENTRY/derived_parameters-group
- /NXellipsometry/ENTRY/derived_parameters/depolarization-field
- /NXellipsometry/ENTRY/experiment_description-field
- /NXellipsometry/ENTRY/experiment_identifier-field
- /NXellipsometry/ENTRY/INSTRUMENT-group
- /NXellipsometry/ENTRY/INSTRUMENT/angle_of_incidence-field
- /NXellipsometry/ENTRY/INSTRUMENT/angular_spread-field
- /NXellipsometry/ENTRY/INSTRUMENT/calibration-group
- /NXellipsometry/ENTRY/INSTRUMENT/calibration/calibration_data-group
- /NXellipsometry/ENTRY/INSTRUMENT/calibration/calibration_data/calibration_angle_of_incidence-field
- /NXellipsometry/ENTRY/INSTRUMENT/calibration/calibration_data/calibration_data-field
- /NXellipsometry/ENTRY/INSTRUMENT/calibration/calibration_data/calibration_data_type-field
- /NXellipsometry/ENTRY/INSTRUMENT/calibration/calibration_data/calibration_wavelength-field
- /NXellipsometry/ENTRY/INSTRUMENT/calibration/calibration_sample-field
- /NXellipsometry/ENTRY/INSTRUMENT/calibration/calibration_time-field
- /NXellipsometry/ENTRY/INSTRUMENT/calibration_status-field
- /NXellipsometry/ENTRY/INSTRUMENT/company-field
- /NXellipsometry/ENTRY/INSTRUMENT/construction_year-field
• /NXellipsometry/ENTRY/INSTRUMENT/data_correction-field
• /NXellipsometry/ENTRY/INSTRUMENT/DETECTOR-group
• /NXellipsometry/ENTRY/INSTRUMENT/DETECTOR/detector_type-field
• /NXellipsometry/ENTRY/INSTRUMENT/DETECTOR/fixed_revolution-field
• /NXellipsometry/ENTRY/INSTRUMENT/DETECTOR/intensity_threshold-field
• /NXellipsometry/ENTRY/INSTRUMENT/DETECTOR/min_intensity-field
• /NXellipsometry/ENTRY/INSTRUMENT/DETECTOR/other_detector-field
• /NXellipsometry/ENTRY/INSTRUMENT/DETECTOR/revolution-field
• /NXellipsometry/ENTRY/INSTRUMENT/DETECTOR/rotating_element-field
• /NXellipsometry/ENTRY/INSTRUMENT/DETECTOR/variable_revolution-field
• /NXellipsometry/ENTRY/INSTRUMENT/ellipsometry_type-field
• /NXellipsometry/ENTRY/INSTRUMENT/firmware-field
• /NXellipsometry/ENTRY/INSTRUMENT/firmware@url-attribute
• /NXellipsometry/ENTRY/INSTRUMENT/firmware@version-attribute
• /NXellipsometry/ENTRY/INSTRUMENT/focussing_probes-field
• /NXellipsometry/ENTRY/INSTRUMENT/tight_source-group
• /NXellipsometry/ENTRY/INSTRUMENT/model-field
• /NXellipsometry/ENTRY/INSTRUMENT/model@version-attribute
• /NXellipsometry/ENTRY/INSTRUMENT/spectrometer-group
• /NXellipsometry/ENTRY/INSTRUMENT/spectrometer/GRATING-group
• /NXellipsometry/ENTRY/INSTRUMENT/spectrometer/GRATING/angular_dispersion-field
• /NXellipsometry/ENTRY/INSTRUMENT/spectrometer/GRATING/grating_wavelength_max-field
• /NXellipsometry/ENTRY/INSTRUMENT/spectrometer/GRATING/grating_wavelength_min-field
• /NXellipsometry/ENTRY/INSTRUMENT/spectrometer/SLIT-group
• /NXellipsometry/ENTRY/INSTRUMENT/spectrometer/SLIT/fixed_slit-field
• /NXellipsometry/ENTRY/INSTRUMENT/spectrometer/SLIT/max_gap-field
• /NXellipsometry/ENTRY/INSTRUMENT/spectrometer/spectral_resolution-field
• /NXellipsometry/ENTRY/INSTRUMENT/spectrometer/wavelength-field
• /NXellipsometry/ENTRY/INSTRUMENT/stage-group
• /NXellipsometry/ENTRY/INSTRUMENT/stage/description-field
• /NXellipsometry/ENTRY/INSTRUMENT/stage/stage_type-field
• /NXellipsometry/ENTRY/INSTRUMENT/stage/TRANSFORMATIONS-group
• /NXellipsometry/ENTRY/INSTRUMENT/stage/TRANSFORMATIONS/alternative-field
• /NXellipsometry/ENTRY/INSTRUMENT/window-group
• /NXellipsometry/ENTRY/INSTRUMENT/window/material-field
• /NXellipsometry/ENTRY/INSTRUMENT/window/orientation_angle-field
• /NXellipsometry/ENTRY/INSTRUMENT/window/other_material-field
• /NXellipsometry/ENTRY/INSTRUMENT/window/reference_data-group
• /NXellipsometry/ENTRY/INSTRUMENT/window/reference_data/data-field
• /NXellipsometry/ENTRY/INSTRUMENT/window/reference_data/reference_sample-field
• /NXellipsometry/ENTRY/INSTRUMENT/window/reference_data/reference_wavelength-field
• /NXellipsometry/ENTRY/INSTRUMENT/window/thickness-field
• /NXellipsometry/ENTRY/operator-group
• /NXellipsometry/ENTRY/operator/address-field
• /NXellipsometry/ENTRY/operator/affiliation-field
• /NXellipsometry/ENTRY/operator/email-field
• /NXellipsometry/ENTRY/operator/name-field
• /NXellipsometry/ENTRY/operator/orcid-field
• /NXellipsometry/ENTRY/operator/telephone_number-field
• /NXellipsometry/ENTRY/plot-group
• /NXellipsometry/ENTRY/plot@axes-attribute
• /NXellipsometry/ENTRY/SAMPLE-group
• /NXellipsometry/ENTRY/SAMPLE/atom_types-field
• /NXellipsometry/ENTRY/SAMPLE/column_names-field
• /NXellipsometry/ENTRY/SAMPLE/data_error-field
• /NXellipsometry/ENTRY/SAMPLE/data_identifier-field
• /NXellipsometry/ENTRY/SAMPLE/data_type-field
• /NXellipsometry/ENTRY/SAMPLE/environment_conditions-group
• /NXellipsometry/ENTRY/SAMPLE/environment_conditions/medium-field
• /NXellipsometry/ENTRY/SAMPLE/environment_conditions/medium_refractive_indices-field
• /NXellipsometry/ENTRY/SAMPLE/environment_conditions/number_of_runs-field
• /NXellipsometry/ENTRY/SAMPLE/environment_conditions/optical_excitation-group
• /NXellipsometry/ENTRY/SAMPLE/environment_conditions/optical_excitation/broadening-field
• /NXellipsometry/ENTRY/SAMPLE/environment_conditions/optical_excitation/duration-field
• /NXellipsometry/ENTRY/SAMPLE/environment_conditions/optical_excitation/pulse_energy-field
• /NXellipsometry/ENTRY/SAMPLE/environment_conditions/optical_excitation/wavelength-field
• /NXellipsometry/ENTRY/SAMPLE/environment_conditions/SENSOR-group
• /NXellipsometry/ENTRY/SAMPLE/environment_conditions/varied_parameters-field
• /NXellipsometry/ENTRY/SAMPLE/layer_structure-field
• /NXellipsometry/ENTRY/SAMPLE/measured_data-field
• /NXellipsometry/ENTRY/SAMPLE/preparation_date-field
• /NXellipsometry/ENTRY/SAMPLE/sample_history-field
nxem

Status:

application definition, extends NXobject

Description:

Characterization and session with one sample in an electron microscope.

**The idea and aim of nxem:** Electron microscopes (EM), whether it be a scanning electron microscope (SEM) or a transmission electron microscope (TEM), are versatile tools for preparing and characterizing samples and specimens. The term specimen is here understood as a synonym for a sample. A specimen is a physical portion of material that is studied/characterize in the microscope session, eventually in different places on the specimen surface. These places are named regions of interest (ROIs).

Fundamentally, an EM is an electron accelerator. Experimentalists use an EM in sessions during which they characterize as well as prepare specimens. This application definition describes data and metadata about processes and characterization tasks applied to one specimen.

Multiple specimens have to be described with multiple NXentry instances.

There are research groups who use an EM in a manner where it is exclusively operated by a single, instrument-responsible scientists or a team of (staff) scientists. These users perform analyses for other users as a service task. Oftentimes, though, and especially for cutting-edge instruments, the scientists and their team guide the process while operating the microscope. Oftentimes the scientists operate the instrument themselves either on-site or remotely and can ask technicians for support. In all cases, these people are considered users. Users might have different roles though.

The rational behind a common EM schema rather than separate SEM or TEM schemata are primarily the key similarities of SEM and TEM instruments: Both have electro-magnetic lenses. These lens may differ in design, alignment, number, and level of corrected-for aberrations. As an obvious difference, a TEM is used mainly to measure the transmitted electron beam. This demands thinner specimens as in SEM but offers capabilities for probing of additional physical mechanisms of electron-matter interaction.

Compared to SEMs, TEMs have a different relative arrangement between the lenses and the specimen which is most obvious by the different relative arrangement of the objective lens versus the specimen.

Nevertheless, both types of electron microscopes use detector systems which measure different types of signals that originate though from the same set of radiation/specimen interactions. Consequently, detectors can also be similar.

Given these physical and technical differences, different instruments have been developed. This led to a coexistence of two broad interacting communities: SEM and TEM users. From a data science perspective, we acknowledge that the more specific a research question is and the narrower the addressed user base is which develops or uses schemata for research data management with EM, the more understandable it is that scientists of either community (or sub-community) ask for method-specific schemata.

Researchers who have a single (main) microscope of some vendor in their lab, may argue they need an NXem_vendor_name schema or an NXem_microscope_name or an NXem_sem or a NXem_tem schema. Scientists exclusively working with one technique or type of signal probed (X-rays, electrons) may argue they wish to be pragmatic and store only what is immediately relevant for their particular technique.
and research questions. In effect, they may advocate for method-specific schemata such as NXem_ebsd, NXem_eels, NXem_edx, or NXem_imaging.

The development in the past has shown that these activities led to a zoo of schemata and implementations of these into many data and file formats. There is nothing which prevents the communities to make these schemata open and interoperable. Open here means specifically not that all data are compliant with/or use the schema and have to end up in the open-source domain. There can be embargo periods first of all. Open means that the metadata and associated schemata are documented in a manner that as many details as possible are open in the sense that others can understand what the (meta)data mean conceptually. The FAIR principles guide all decisions how data and metadata should be stored.

EM instruments, software, and research are moving targets. Consequently, there is a key challenge and inconvenience with having many different schemata with associated representations of data and metadata in EM: Each combination of schemata or an interoperable-made handshake between two file formats or software packages has to be maintained by software developers. This counts especially when data should be processed interoperably between software packages.

This brings two problems: Many software tools and parsers for the handshaking between tools to maintain. This can result in usage of different terminology. Which in turn results in representations and connections made between different data representations and workflows that are not machine-actionable. There are community efforts to harmonize the terminology.

A common vocabulary can serve interoperability as developers of schemata and scientists can take for instance then these terms as closely as possible. Ideally, they specialize the application definition only for the few very specific additional quantities of their instruments and techniques. This is better than reimplementing the wheel for descriptions of EM instruments. This route of more standardization can support the EM community in that it removes the necessity for having to maintain a very large number of schemata.

Aiming for more standardization, i.e. a lower number of schemata rather than a single standard for electron microscopy is a compromise that can serve academia as it enables the EM community to focus their software development efforts on those schemata, on fixing and discussing them, and on harmonize their common vocabulary. These activities can be specifically relevant also for vendors of EM hard- and software as it improves the longevity of certain schema and thus can help to incentivize vendors to support the community with implementing support for such schemata into their proprietary applications.

In effect, everybody can gain from this as it will likely reduce the cases in which scientists have to fix bugs in making their own tools compliant and interoperable with tools of their colleagues and the wider community.

The here proposed NXem application definition offers modular components (EM-research specific base classes) for using NeXus to define schemata for electron microscopy research. Working towards a common vocabulary is a community activity that profits from everybody reflecting in detail whether certain terms they have used are not eventually conceptually similar if not the same as what this application definition and its base classes provide.

We are happy for receiving your feedback.

It is noteworthy to understand that (not only for) NeXus, schema differ already if at least one field is required in one version of the schema, but it is set optional in another version. If group(s), field(s), or attributes are removed or added, or even a docstring is changed, schemata can become inconsistent. An application definition here serves as a contract between a data provider and a data consumer. These two can be software tools (like the vendor software to drive the instrument or a scientific software for doing artificial intelligence with EM data). Such changes of a schema lead to new versions.

Tools like NeXus do not avoid or protect against inconsistencies; however NeXus offers a mechanism and toolset, through which schemata can be documented and defined. In effect, having an openly documented (at a case-specific level of technical detail) schema is a necessary but alone not a sufficient step to take EM research on a route of machine-actionable and interoperable FAIR data. A common vocabulary and
a machine-actionable knowledge representation/engine is also required. Essentially when the docstrings are no longer needed but can be replaced by a connection to an automated tool which understands what a specific field represents conceptually, EM data have become more generally interoperable EM data.

This application definition takes a key step into this direction. It offers a controlled vocabulary and relation between concepts and data relevant for research with electron microscopes. To be most efficient and offering reusability, the application definition should be understood as a template that one should ideally use as is. This application definition is called NXem. It can be considered a base for more specialized definitions (ideally prefixed with NXem) method.

The use of NXem should be as follows: Offspring application definitions should not remove groups but make them optional or, even better, propose changes in the application definition.

A particular challenge with electron microscopes as physical instruments are their dynamics. To make EM data understandable, repeatable, and eventually corresponding experiments reproducible in general requires a documentation of the spatio-temporal dynamics of the instrument in its environment. For most commercial systems there is a specific accessibility beyond which detailed settings like lens excitations and low-level hardware settings may not be retrievable.

EM experiments by design illuminate the specimen with electrons as a consequence of which the specimen changes if not may get destroyed. As such, repeatability of numerical processing and clear descriptions of procedures and system setups should be addressed first.

If especially a certain simulation package needs a detailed view of the geometry of the lens system and its excitations during the course of the experiment, it is difficult to fully abstract the technical details of the hardware into a set of names for fields and groups that make for a compromise between clarity and being vendor-agnostic. Settings of apertures are an example where aperture modes are aliases behind which there is a set of settings. These settings are difficult to retrieve, often undocumented in detail. This serves users and makes EM experiments easier understandable and conveniently executable for a broader user base. The opportunities for application definitions to offer an abstraction layer are limited.

Instead, currently it is for the docstring to specify what is conceptually eventually behind such aliases. The design rule we followed while drafting the application definition and base classes is that there are numerous (technical) details about an EM which may warrant a very detailed technical disentangling of settings and reflection of numerous settings as deeply nested groups, fields and attributes. An application definition can offer a place to hold these nested representations; however at the cost of generality.

Which specific details matter for answering scientific research questions is a difficult question to answer by a single team of scientists, especially if the application definition is to speak for a number of vendors. What makes it especially challenging if the application definition is expected to hold all data that might be of relevance for future questions.

We are skeptical if there is one representation that can fulfill all these aims, while remaining at the same time approachable and executable by a large number of scientists in a community. With this application definition we would like to motivate the community to work towards such aim. While doing so we found that existent terminology can be encoded into a more controlled vocabulary.

We have concluded that despite all these details of current EM research with SEM, TEM, and focused-ion beam instruments, there a clearly identifiable common components and generalizable settings of EM research use cases.

This application definition has the following components at the top-level:

- Generic experimental details (timestamp, identifiers, name); conceptually these are session details. A session at a microscope may involv the characterization of multiple specimens. For each specimen an instance of an (NXentry) is created. Details of the instrument have to be stored at least in an entry. Other entries should refer to these metadata via links to reduce redundancies.
- Each signal, such as a spectrum or image taken at the microscope, should have an associated time stamp and report of the specific settings at that point in time when the image was taken. The reason is
that EMs can be highly dynamic, be used to illuminate the specimen differently or show drift during signal acquisition, to name but a few effects. What constitutes a single EM experiment/measurement? This can be the collecting of a single diffraction pattern with a scanning TEM (STEM), taking of a secondary electron image for fracture analysis, taking a set of EBSD line scan and surface mappings in an SEM, or ion-beam-milling of a specimen in preparation for an atom probe experiment.

- NXmonitor; instances to keep track of time-dependent quantities pertaining to specific components of the instrument. Alternatively NXevent_data_em instances can be used to store timestamp states of the components, which is relevant to document the exact settings when images and spectra were taken.

- NXinstrument; conceptually this is a container to store arbitrary level of detail of the technical components of the microscope as a device and the lab in which it is operated.

- NXuser; conceptually, this is a set with at least one NXuser instance which details who operated or performed the measurement. Additional NXusers can be referred to in an NXevent_data_em instance to store individualized details who executed an event.

- NXevent_data_em instances as an NXevent_data_em_set; each NXevent_data_em instance is a container to group specific details about the state of the microscope when a measurement was taken and relevant data and eventual processing steps were taken (on-the-fly).

- NXdata; a the top-level, conceptually, this is a place for documenting available default plottable data. A default plottable can be useful for research data management systems to show a visual representation of some aspect of the content of the EM session. It is clear that what constitutes a useful default plot is a matter of interpretation, somewhat of personal taste, and community standards.

In effect, default plottables are case- and method-specific. Usually a session at a microscope is used to collect multiple signals and images. Examples for possible default plottables could be arbitrarily taken: secondary, back-scattered, electron image, diffraction pattern, EELS spectra, composition, or orientation mappings to name but a few.

There are a few design choices to consider with sub-ordinate groups:

- Above images, spectra, and mappings should be stored as NXdata instances, ideally formatted in such a way that they can be displayed with visualization software that can be specific for the file format in which the data are stored. NeXus specifies only the data model, i.e. the terms and their relations. These descriptions can be implemented and stored in JSON, HDF5, XML, or HSDS, file storage, or even other formats, although HDF5 is the most commonly used.

- Consumable results of EM characterization tasks are usually a sub-set of data artifacts, as there is not an infinite amount of possible electron/ion beam-specimen interactions.

- Images of electron counts detected in specific operation modes (bright field, dark field in TEM, secondary/back-scattered, Kikuchi in SEM)

- Spectra (X-ray quanta or auger electron counts)

- These data are in virtually all cases a result of some numerical processing. It makes sense to name them with a controlled vocabulary, e.g. SE (secondary electron), BSE (back-scattered electron), Kikuchi, X-ray, Auger, Cathodoluminescence etc.

A key question often asked with EM experiments is how the actual (meta)data should be stored (in memory or on disk). To this end the schema, here makes no specific assumptions, not even that all the fields/group of a schema instance have to be stored into a single file. Instead, the schema specifies the relations between metadata, constraints on how they should be formatted, what they conceptually represent and which terms (controlled vocabulary) is practical to store with the data.

In effect, the application definition is a graph which describes how (meta)data are related to one another.

Symbols:
No symbol table

Groups cited:

NXcoordinate_system_set, NXdata, NXdetector, NXbeam_column, NXentry, NXevent_data_em_set, NXevent_data_em, NXibeam_column, NXimage_set_em_adf, NXimage_set_em_bf, NXimage_set_em_bse, NXimage_set_em_chamber, NXimage_set_em_df, NXimage_set_em_diffrac, NXimage_set_em_ecci, NXimage_set_em_kikuchi, NXimage_set_em_ronchigram, NXimage_set_em_se, NXinstrument, NXmanufacturer, NXmonitor, NXnote, NOptical_system_em, NPump, NSample, NScanbox_em, NSpectrum_set_em_auger, NSpectrum_set_em_cathodolum, NSpectrum_set_em_eels, NSpectrum_set_em_xray, NXuser

Structure:

ENTRY: (required) NXentry

@version: (required) NX_CHAR

An at least as strong as SHA256 hashvalue of the file that specifies the application definition.

definition: (required) NX_CHAR

NeXus NXDL schema to which this file conforms.

Obligatory value: NXem

experiment_identifier: (required) NX_CHAR

Ideally, a (globally) unique persistent identifier for referring to this experiment.

The identifier is usually defined/issued by the facility, laboratory, or the principle investigator. The identifier enables to link experiments to e.g. proposals.

experiment_description: (optional) NX_CHAR

Free-text description about the experiment.

Users are strongly advised to detail the sample history in the respective field and fill rather as completely as possible the fields of this application definition rather than write details about the experiment into this free-text description field.

start_time: (required) NX_DATE_TIME

ISO 8601 time code with local time zone offset to UTC information included when the microscope session started. If the application demands that time codes in this section of the application definition should only be used for specifying when the experiment was performed - and the exact duration is not relevant - this start time field should be used.

Often though it is useful to specify a time interval with specifying both start_time and end_time to allow for more detailed bookkeeping and interpretation of the experiment. The user should be aware that even with having both time instances specified, it may not be possible to infer how long the experiment took or for how long data were acquired.

More detailed timing data over the course of the experiment have to be collected to compute this. These computations can take advantage of individual time stamps in NXevent_em instances to provide additional pieces of information.

end_time: (required) NX_DATE_TIME

ISO 8601 time code with local time zone offset to UTC included when the microscope session ended.

program: (required) NX_CHAR
Commercial or otherwise given name to the program which was used to create the file.

Electron microscopy experiments are usually controlled/performed via commercial integrated acquisition and instrument control software. In many cases, an EM dataset is useful only if it gets post-processed already during the acquisition, i.e. while the scientist is sitting at the microscope. Many of these processes are automated, while some demand GUI interactions with the control software. Examples include collecting of diffraction pattern and on-the-fly indexing of these.

It is possible that different types of programs might be used to perform these processing steps whether on-the-fly or not. If this is the case the processing should be structured with individual NXprocess instances. If the program and/or version used for processing referred to in an NXprocess group is different to the program and version mentioned in this field, the NXprocess needs to hold an own program and version.

@version: (required) NX_CHAR

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

experiment_documentation: (optional) NXnote

Binary container for a file or a compressed collection of files which can be used to add further descriptions and details to the experiment. The container can hold a compressed archive.

thumbnail: (optional) NXnote

A small image that is representative of the entry; this can be an image taken from the dataset like a thumbnail of a spectrum. A 640 x 480 pixel jpeg image is recommended. Adding a scale bar to that image is recommended but not required as the main purpose of the thumbnail is to provide e.g. thumbnail images for displaying them in data repositories.

@type: (required) NX_CHAR

operator: (required) NXuser

Contact information and eventually details of at least one person involved in the taking of the microscope session. This can be the principle investigator who performed this experiment. Adding multiple users if relevant is recommended.

name: (required) NX_CHAR

Given (first) name and surname of the user.

affiliation: (recommended) NX_CHAR

Name of the affiliation of the user at the point in time when the experiment was performed.

address: (recommended) NX_CHAR

Postal address of the affiliation.

email: (required) NX_CHAR

Email address of the user at the point in time when the experiment was performed. Writing the most permanently used email is recommended.

orcid: (recommended) NX_CHAR
Globally unique identifier of the user as offered by services like ORCID or ResearcherID.

**telephone_number**: (optional) *NX_CHAR*

(Business) telephone number of the user at the point in time when the experiment was performed.

**role**: (optional) *NX_CHAR*

Which role does the user have in the place and at the point in time when the experiment was performed? Technician operating the microscope. Student, postdoc, principle investigator, guest are common examples.

**social_media_name**: (optional) *NX_CHAR*

Account name that is associated with the user in social media platforms.

**social_media_platform**: (optional) *NX_CHAR*

Name of the social media platform where the account under social_media_name is registered.

**SAMPLE**: (required) *NXsample*

A description of the material characterized in the experiment. Sample and specimen are threaded as de facto synonyms.

**method**: (required) *NX_CHAR*

A qualifier whether the sample is a real one or a virtual one (in a computer simulation)

Any of these values: experimental | simulation

**name**: (required) *NX_CHAR*

Descriptive name or ideally (globally) unique persistent identifier. The name distinguishes the specimen from all others and especially the predecessor/origin from where the specimen was cut.

This field must not be used for an alias of the sample. Instead, use short_title.

In cases where multiple specimens have been loaded into the microscope the name has to identify the specific one, whose results are stored by this NXentry, because a single NXentry should be used only for the characterization of a single specimen.

Details about the specimen preparation should be stored in the sample history.

**sample_history**: (required) *NX_CHAR*

Ideally, a reference to a (globally) unique persistent identifier, representing a data artifact which documents ideally as many details of the material, its microstructure, and its thermo-chemo-mechanical processing/preparation history as possible.

The sample_history is the record what happened before the specimen was placed into the microscope at the beginning of the session.

In the case that such a detailed history of the sample/specimen is not available, use this field as a free-text description to specify a sub-set of the entire sample history, i.e. what you would consider are the key steps and relevant information about the specimen, its material, microstructure, thermo-chemo-mechanical processing state, and the details of the preparation.
Specific details about eventual physically-connected material like embedding resin should be documented ideally also in the sample_history. If all fails, the description field can be used but it is strongly discouraged because it leads to eventually non-machine-actionable data.

**preparation_date**: (required) *NX_DATE_TIME*

ISO 8601 time code with local time zone offset to UTC information when the specimen was prepared.

Ideally report the end of the preparation, i.e. the last known time the measured specimen surface was actively prepared. Usually this should be a part of the sample history, i.e. the sample is imagined handed over for the analysis. At the point it enters the microscope the session starts.

Knowing when the specimen was exposed to e.g. specific atmosphere is especially required for environmentally sensitive material such as hydrogen charged specimens or experiments including tracers with a short half time. Further time stamps prior to preparation_date should better be placed in resources which describe the sample_history.

**short_title**: (optional) *NX_CHAR*

Possibility to give an abbreviation or alias of the specimen name field.

**atom_types**: (required) *NX_CHAR*

Use Hill’s system for listing elements of the periodic table which are inside or attached to the surface of the specimen and thus relevant from a scientific point of view.

The purpose of the field is to offer materials database systems an opportunity to parse the relevant elements without having to interpret these from the sample history.

**thickness**: (required) *NX_FLOAT* {units=*NX_LENGTH*}

(Measured) sample thickness. The information is recorded to qualify if the beam used was likely able to shine through the specimen.

**description**: (optional) *NX_CHAR*

Discouraged free-text field in case properly designed records for the sample_history are not available.

**DATA**: (required) *NXdata*

Hard link to a location in the hierarchy of the NeXus file where the data for default plotting are stored.

**COORDINATE_SYSTEM_SET**: (required) *NXcoordinate_system_set*

**MONITOR**: (optional) *NXmonitor*

**em_lab**: (required) *NXinstrument*

Metadata and numerical data of the microscope and the lab in which it stands.

The em_lab section contains a description of the instrument and its components. The component descriptions in this section differ from those inside individual NX-event_em sections. These event instances take the role of time snapshot. For an NX-event_em instance users should store only those settings for a component which are relevant to understand the current state of the component. Here, current means at the point in time, i.e. the time interval, which the event represents.
For example, it is not relevant to store in each event's electron_gun group again the
details of the gun type and manufacturer but only the high-voltage if for that event
the high-voltage was different. If for all events the high-voltage was the same it is not
even necessary to include an electron_gun section in the event.

Individual sections of specific type should have the following names:

- **NXaperture**: the name should match with the name of the lens
- **NXlens_em**: condenser_lens, objective_lens are commonly used names
- **NXcorrector_cs**: device for correcting spherical aberrations
- **NXstage_lab**: a collection of component for holding the specimen and eventual
  additional component for applying external stimuli on the sample
- **NXdetector**: several possible names like secondary_electron, backscattered_electron, direct_electron, ebsd, edx, wds, auger, cathodoluminescence, camera, ronchigram

**instrument_name**: (required) **NX_CHAR**

Given name of the microscope at the hosting institution. This is an alias.
Examples could be NionHermes, Titan, JEOL, Gemini, etc.

**location**: (optional) **NX_CHAR**

Location of the lab or place where the instrument is installed. Using GEOREF
is preferred.

**MANUFACTURER**: (required) **NXmanufacturer**

**EBEAM_COLUMN**: (required) **NXebeam_column**

**IBEAM_COLUMN**: (optional) **NXibeam_column**

**ebeam_deflector**: (required) **NXscanbox_em**

**ibeam_deflector**: (optional) **NXscanbox_em**

**OPTICAL_SYSTEM_EM**: (optional) **NXoptical_system_em**

**DETECTOR**: (required) **NXdetector**

- Description of the type of the detector.

  Electron microscopes have typically multiple detectors. Different technolo-
gies are in use like CCD, scintillator, direct electron, CMOS, or image plate
to name but a few.

  **description**: (optional) **NX_CHAR**

  Free text option to write further details about the detector.

**MANUFACTURER**: (required) **NXmanufacturer**

**PUMP**: (optional) **NXpump**

**measurement**: (optional) **NXevent_data_em_set**

A container to structure a set of NXevent_em instances.

An event is a time point/interval during which the microscope was configured in a
specific way and the microscope was used to take a measurement.
Each NXevent_em holds an acquisition task with the microscope. For instance the capturing of a secondary electron, backscattered electron, diffraction image, or spectrum.

An NXevent_em_data instance holds specific details about how raw data from a detector were processed into consumable data like images, spectra, etc. These on-the-fly data processing tasks are usually performed by the control software, eventually realized with custom scripts.

Furthermore, NXevent_em_state instances can document specific values and settings of the microscope during the snapshot/event.

**EVENT_DATA_EM: (required) NXevent_data_em**

A container holding a specific result of the measurement and eventually metadata how that result was obtained numerically.

NXevent_em instances can hold several specific NXimage_em or NXspectrum_em instances taken and considered as one event, i.e. a point in time when the microscope had the settings specified either in NXinstrument or in this NXevent_data_em instance.

The application definition is designed without an explicit need an NXevent_data_em instance that contains an NXimage_em or NXspectrum_em instance. An NXevent_data_em can be used to document a specific state of the microscope at a time without having it placed into the NXinstrument group.

In other words the NXinstrument group details primarily the more static settings and components of the microscope as they are found by the operator during the session. The NXevent_data_em samples the dynamics.

It is not necessary to store data in NXebeam, NXibeam instances of NXevent_data_em but in this case it is assumed that the settings were constant over the entire course of microscope session and thus all relevant metadata inside the NXinstrument groups are sufficient to understand the session.

**start_time: (required) NX_DATE_TIME**

**end_time: (required) NX_DATE_TIME**

**event_identifier: (required) NX_CHAR**

Reference to a specific state and setting of the microscope components.

**event_type: (required) NX_CHAR**

**detector_identifier: (required) NX_CHAR**

The detector or set of detectors that was used to collect this signal. The name of the detector has to match one of the names of available NXdetector instances e.g. if the instrument has an ebsd_camera the detector for an NXimage_em_kikuchi should be the NXdetector instance called ebsd_camera.

**IMAGE_SET_EM_SE: (optional) NXimage_set_em_se**

**IMAGE_SET_EM_BSE: (optional) NXimage_set_em_bse**

**IMAGE_SET_EM_ECCI: (optional) NXimage_set_em_ecci**

**IMAGE_SET_EM_BF: (optional) NXimage_set_em_bf**

**IMAGE_SET_EM_DF: (optional) NXimage_set_em_df**
**IMAGE_SET_EM_ADF**: (optional) \textit{NXimage_set_em_adf}

**IMAGE_SET_EM_KIKUCHI**: (optional) \textit{NXimage_set_em_kikuchi}

**IMAGE_SET_EM_DIFFRAC**: (optional) \textit{NXimage_set_em_diffrac}

**SPECTRUM_SET_EM_XRAY**: (optional) \textit{NXspectrum_set_em_xray}

**SPECTRUM_SET_EM_EELS**: (optional) \textit{NXspectrum_set_em_eels}

**SPECTRUM_SET_EM_AUGER**: (optional) \textit{NXspectrum_set_em_auger}

**SPECTRUM_SET_EM_CATHODOLUM**: (optional) \textit{NXspectrum_set_em_cathodolum}

**IMAGE_SET_EM_RONCHIGRAM**: (optional) \textit{NXimage_set_em_ronchigram}

**IMAGE_SET_EM_CHAMBER**: (optional) \textit{NXimage_set_em_chamber}

**EBEAM_COLUMN**: (optional) \textit{NXebeam_column}

**IBEAM_COLUMN**: (optional) \textit{NXibeam_column}

**ebeam_deflector**: (optional) \textit{NXscanbox_em}

**ibeam_deflector**: (optional) \textit{NXscanbox_em}

**OPTICAL_SYSTEM_EM**: (optional) \textit{NXoptical_system_em}

**USER**: (optional) \textit{NXuser}

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- \textit{/NXem/ENTRY-group}
- \textit{/NXem/ENTRY/COORDINATE_SYSTEM_SET-group}
- \textit{/NXem/ENTRY/DATA-group}
- \textit{/NXem/ENTRY/definition-field}
- \textit{/NXem/ENTRY/em_lab-group}
- \textit{/NXem/ENTRY/em_lab/DETECTOR-group}
- \textit{/NXem/ENTRY/em_lab/DETECTOR/description-field}
- \textit{/NXem/ENTRY/em_lab/DETECTOR/MANUFACTURER-group}
- \textit{/NXem/ENTRY/em_lab/EBEAM_COLUMN-group}
- \textit{/NXem/ENTRY/em_lab/ebeam_deflector-group}
- \textit{/NXem/ENTRY/em_lab/IBEAM_COLUMN-group}
- \textit{/NXem/ENTRY/em_lab/ibeam_deflector-group}
- \textit{/NXem/ENTRY/em_lab/instrument_name-field}
- \textit{/NXem/ENTRY/em_lab/location-field}
- \textit{/NXem/ENTRY/em_lab/MANUFACTURER-group}
- \textit{/NXem/ENTRY/em_lab/OPTICAL_SYSTEM_EM-group}
• /NXem/ENTRY/em_lab/PUMP-group
• /NXem/ENTRY/end_time-field
• /NXem/ENTRY/experiment_description-field
• /NXem/ENTRY/experiment_documentation-group
• /NXem/ENTRY/experiment_identifier-field
• /NXem/ENTRY/measurement-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/detector_identifier-field
• /NXem/ENTRY/measurement/EVENT_DATA_EM/EBEAM_COLUMN-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/ebeam_deflector-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/end_time-field
• /NXem/ENTRY/measurement/EVENT_DATA_EM/event_identifier-field
• /NXem/ENTRY/measurement/EVENT_DATA_EM/event_type-field
• /NXem/ENTRY/measurement/EVENT_DATA_EM/IBEAM_COLUMN-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/ibeam_deflector-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET_EM_ADF-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET_EM_BF-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET_EM_BSE-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET_EM_CHAMBER-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET_EM_DF-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET_EM_DIFFRAC-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET_EM_ECCI-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET_EM_KIKUCHI-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET_EM_RONCHIGRAM-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET_EM_SE-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/OPTICAL_SYSTEM_EM-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET_EM_AUGER-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET_EM_CATHODOLUM-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET_EM_EELS-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET_EM_XRAY-group
• /NXem/ENTRY/measurement/EVENT_DATA_EM/start_time-field
• /NXem/ENTRY/measurement/EVENT_DATA_EM/USER-group
• /NXem/ENTRY/MONITOR-group
• /NXem/ENTRY/operator-group
• /NXem/ENTRY/operator/address-field
• /NXem/ENTRY/operator/affiliation-field
NXenergydispersion

Status:

base class, extends NXobject

Description:

Subclass of NXelectronanalyser to describe the energy dispersion section of a photoelectron analyser.

Symbols:

No symbol table

Groups cited:

NXaperture, NXdeflector, NXlens_em

Structure:

scheme: (optional) NX_CHAR

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXem.nxdl.xml
Energy dispersion scheme employed, for example: tof, hemispherical, cylindrical, mirror, retarding grid, etc.

**pass_energy**: (optional) *NX_FLOAT* {units=NX\_ENERGY}

Energy of the electrons on the mean path of the analyser. Pass energy for hemispherics, drift energy for tofs.

**center_energy**: (optional) *NX_FLOAT* {units=NX\_ENERGY}

Center of the energy window

**energy_interval**: (optional) *NX_FLOAT* {units=NX\_ENERGY}

The interval of transmitted energies. It can be two different things depending on whether the scan is fixed or swept. With a fixed scan it is a 2 vector containing the extrema of the transmitted energy window (smaller number first). With a swept scan of \( m \) steps it is a 2x\( m \) array of windows one for each measurement point.

**diameter**: (optional) *NX_FLOAT* {units=NX\_LENGTH}

Diameter of the dispersive orbit

**energy_scan_mode**: (optional) *NX_CHAR*

Way of scanning the energy axis (fixed or sweep).

Any of these values: fixed | sweep

**tof_distance**: (optional) *NX_FLOAT* {units=NX\_LENGTH}

Length of the tof drift electrode

**APERTURE**: (optional) *NXaperture*

Size, position and shape of a slit in dispersive analyzer, e.g. entrance and exit slits.

**DEFLECTOR**: (optional) *NXdeflector*

Deflectors in the energy dispersive section

**LENS\_EM**: (optional) *NXlens\_em*

Individual lenses in the energy dispersive section

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXenergydispersion/APERTURE-group
- /NXenergydispersion/center_energy-field
- /NXenergydispersion/DEFLECTOR-group
- /NXenergydispersion/diameter-field
- /NXenergydispersion/energy_interval-field
- /NXenergydispersion/energy_scan_mode-field
- /NXenergydispersion/LENS\_EM-group
- /NXenergydispersion/pass_energy-field
- /NXenergydispersion/scheme-field
NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXenergydispersion.nxdl.xml

NXevent_data_em

Status:
base class, extends NXobject

Description:

Metadata and settings of an electron microscope for scans and images.

The need for such a structuring of data is evident from the fact that electron microscopes are dynamic. Oftentimes it suffices to calibrate the instrument at the start of the session. Subsequently, data (images, spectra, etc.) can be collected. Users may wish to take only a single scan or image and complete their microscope session; however frequently users spend much longer at the microscope, recalibrate, and take multiple data items (scans, images, spectra) each coming with own detector and on-the-fly processing settings and calibration.

For the single data item use case one may argue that the need for additional grouping is redundant. Instead, the metadata could equally be stored inside the respective groups of the top-level mandatory NXinstrument group. On the flip side, even for a session with a single image it would also not harm to nest the data.

In fact, oftentimes scientists feel that there is a need to store details about eventual drift of the specimen in its holder (if such data is available) or record changes to the lens excitations caused or apertures used. Although current microscopes are usually equipped with stabilization systems for many of the individual components, it can still be useful to store time-dependent data in detail.

Another reason if not a need for more finely granularizable options for storing time-dependent data, is that over the course of a session one may reconfigure the microscope. What is a reconfiguration? This could be the change of an aperture mode because a scientist may first collect an image with some aperture and then choose a different one. As the aperture affects the electron beam it will affect the system.

Let aside for a moment the technology and business models, an EM could be monitored (and will likely become so more in the future) for streaming out spatio-temporal details about its components, locations of objects and eventually applied stimuli and positioning of the specimen.

Some snapshot or integrated data from this stream are relevant for understanding signal genesis and electron/ion beam paths. In such a generic case it might be necessary to sync these streaming data with those intervals in time when specific measurements are taken (spectra collected, images taken, diffraction images indexed on-the-fly).

Theoretically, an instrument and specimen should be considered as dynamic. Scientists often report or feel (difficult to quantify) observations that microscopes perform differently across sessions, without sometimes being able to identify clear root causes. Users of the instrument may consider such conditions impractical and thus either abort their session or try to bring the microscope first into a state where conditions are considered stable and of high enough quality for collecting data.

In all these cases it is practical to store time-dependent data of the instrument state not in the respective instrument component groups of the top-level NXinstrument but in a sort of a log of event data. This is the idea behind the NXevent_data_em snapshot containers.

The base class requires a start time and an optional end time. The end time should be added to represent a time interval (remind the idea of the instrument state stream) during which the scientist considered the microscope (especially ebeam and specimen) as stable enough.
For specific simulation purposes, mainly in an effort to digitally repeat or simulate the experiment, it is tempting to consider dynamics of the instrument, implemented as time-dependent functional descriptions of e.g. lens excitations, beam shape functions, trajectories of groups of electrons, or detector noise models.

For now the preferred strategy to handle these cases is through customizations of the specific fields within NXevent_data_em instances.

Another alternative could be to sample finer, eventually dissimilarly along the time axis; however this may cause situations where an NXevent_data_em instance does not contain specific measurements (i.e. images, spectra of scientific relevance).

In this case one should better go for a customized application definition with a functional property description inside members (fields or groups) in NXevent_data_em instances or resort to a specific application definition which documents metadata for tracking explicitly electrons (with ray-tracing based descriptors/computational geometry descriptors) or tracking of wave bundles.

This perspective on more subtle time-dependent considerations of electron microscopy can be advantageous also for storing details of time-dependent additional components that are coupled to and/or synced with instrument.

Examples include cutting-edge experiments where the electron beam gets coupled/excited by e.g. lasers. In this case, the laser unit should be registered under the top-level NXinstrument section. Its spatio-temporal details could be stored inside respective groups of the NXinstrument.

Symbols:

No symbol table

Groups cited:

NXbeam_column, NXibeam_column, NXimage_set_em_adf, NXimage_set_em_bf, NXimage_set_em_bse, NXimage_set_em_chamber, NXimage_set_em_df, NXimage_set_em_diffrac, NXimage_set_em_ecci, NXimage_set_em_kikuchi, NXimage_set_em_ronchigram, NXimage_set_em_se, NXoptical_system_em, NXscanbox_em, NXspectrum_set_em_auger, NXspectrum_set_em_cathodolum, NXspectrum_set_em_eels, NXspectrum_set_em_xray, NXuser

Structure:

start_time: (optional) NX_DATE_TIME

ISO 8601 time code with local time zone offset to UTC information included when the snapshot time interval started. If the user wishes to specify an interval of time that the snapshot should represent during which the instrument was stable and configured using specific settings and calibrations, the start_time is the start (left bound of the time interval) while the end_time specifies the end (right bound) of the time interval.

end_time: (optional) NX_DATE_TIME

ISO 8601 time code with local time zone offset to UTC included when the snapshot time interval ended. If the user does not wish to specify a time interval, end_time should have the same value as start_time.

event_identifier: (optional) NX_CHAR

Reference to a specific state and setting of the microscope components.

event_type: (optional) NX_CHAR

Which specific event/measurement type. Examples are:

- In-lens/backscattered electron, usually has quadrants
- Secondary_electron, image, topography, fractography, overview images
- Backscattered_electron, image, Z or channeling contrast (ECCI)
• Bright_field, image, TEM
• Dark_field, image, crystal defects
• Annular dark field, image (medium- or high-angle), TEM
• Diffraction, image, TEM, or a comparable technique in the SEM
• Kikuchi, image, SEM EBSD and TEM diffraction
• X-ray spectra (point, line, surface, volume), composition EDS/EDX(S)
• Electron energy loss spectra for points, lines, surfaces, TEM
• Auger, spectrum, (low Z contrast element composition)
• Cathodoluminescence (optical spectra)
• Ronchigram, image, alignment utility specifically in TEM
• Chamber, e.g. TV camera inside the chamber, education purposes.

detector_identifier: (optional) NX_CHAR

The detector or set of detectors that was used to collect this signal. The name of the detector has to match the names used for available detectors, i.e. if the instrument has an ebsd_camera named detector, instances of NXimage_em_kikuchi should use ebsd_camera as the detector name.

IMAGE_SET_EM_SE: (optional) NXimage_set_em_se
IMAGE_SET_EM_BSE: (optional) NXimage_set_em_bse
IMAGE_SET_EM_ECCI: (optional) NXimage_set_em_ecci
IMAGE_SET_EM_BF: (optional) NXimage_set_em_bf
IMAGE_SET_EM_DF: (optional) NXimage_set_em_df
IMAGE_SET_EM_ADF: (optional) NXimage_set_em_adf
IMAGE_SET_EM_KIKUCHI: (optional) NXimage_set_em_kikuchi
IMAGE_SET_EM_DIFFRAC: (optional) NXimage_set_em_diffrac
SPECTRUM_SET_EM_XRAY: (optional) NXspectrum_set_em_xray
SPECTRUM_SET_EM_EELS: (optional) NXspectrum_set_em_eels
SPECTRUM_SET_EM_AUGER: (optional) NXspectrum_set_em_auger
SPECTRUM_SET_EM_CATHODOLUM: (optional) NXspectrum_set_em_cathodolum
IMAGE_SET_EM_RONCHIGRAM: (optional) NXimage_set_em_ronchigram
IMAGE_SET_EM_CHAMBER: (optional) NXimage_set_em_chamber
EBEAM_COLUMN: (optional) NXebeam_column
IBEAM_COLUMN: (optional) NXibeam_column
ebeam_deflector: (optional) NXscanbox_em
ibeam_deflector: (optional) NXscanbox_em
OPTICAL_SYSTEM_EM: (optional) NXoptical_system_em
USER: (optional) NXuser
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXevent_data_em/detector_identifier-field`
- `/NXevent_data_em/EBEAM_COLUMN-group`
- `/NXevent_data_em/ebeam_deflector-group`
- `/NXevent_data_em/end_time-field`
- `/NXevent_data_em/event_identifier-field`
- `/NXevent_data_em/event_type-field`
- `/NXevent_data_em/IBEAM_COLUMN-group`
- `/NXevent_data_em/ibeam_deflector-group`
- `/NXevent_data_em/IMAGE_SET_EM_ADF-group`
- `/NXevent_data_em/IMAGE_SET_EM_BF-group`
- `/NXevent_data_em/IMAGE_SET_EM_BSE-group`
- `/NXevent_data_em/IMAGE_SET_EM_CHAMBER-group`
- `/NXevent_data_em/IMAGE_SET_EM_DF-group`
- `/NXevent_data_em/IMAGE_SET_EM_DIFFRAC-group`
- `/NXevent_data_em/IMAGE_SET_EM_ECCI-group`
- `/NXevent_data_em/IMAGE_SET_EM_KIKUCHI-group`
- `/NXevent_data_em/IMAGE_SET_EM_RONCHIGRAM-group`
- `/NXevent_data_em/IMAGE_SET_EM_SE-group`
- `/NXevent_data_em/OPTICAL_SYSTEM_EM-group`
- `/NXevent_data_em/SPECTRUM_SET_EM_AUGER-group`
- `/NXevent_data_em/SPECTRUM_SET_EM_CATHODOLUM-group`
- `/NXevent_data_em/SPECTRUM_SET_EM_EELS-group`
- `/NXevent_data_em/SPECTRUM_SET_EM_XRAY-group`
- `/NXevent_data_em/start_time-field`
- `/NXevent_data_em/USER-group`

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXevent_data_em.nxdl.xml
**NXevent_data_em_set**

**Status:**
base class, extends *NXobject*

**Description:**
Container to hold NXevent_data_em instances of an electron microscope session.
An event is a time interval during which the microscope was configured, considered stable, and used for characterization.

**Symbols:**
No symbol table

**Groups cited:**
*NXevent_data_em*

**Structure:**
EVENT_DATA_EM: (optional) *NXevent_data_em*

**Hypertext Anchors**
List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXevent_data_em_set/EVENT_DATA_EM-group

**NXibeam_column**

**Status:**
base class, extends *NXobject*

**Description:**
Container for components of a focused-ion-beam (FIB) system.
FIB capabilities turn especially scanning electron microscopes into specimen preparation labs. FIB is a material preparation technique whereby portions of the sample are illuminated with a focused ion beam with controlled intensity intense enough and with sufficient ion momentum to remove material in a controllable manner.

The fact that an electron microscope with FIB capabilities has needs a second gun with own relevant control circuits, focusing lenses, and other components, warrants an own base class to group these components and distinguish them from the lenses and components for creating and shaping the electron beam.

For more details about the relevant physics and application examples consult the literature, for example:

- L. A. Giannuzzi et al.
- E. I. Preiß et al.
- J. F. Ziegler et al.
- J. Lili

**Symbols:**
No symbol table

Groups cited:
- NXaperture_em
- NXbeam
- NXion
- NXlens_em
- NXmanufacturer
- NXsensor
- NXsource
- NXtransformations

Structure:

**MANUFACTURER**: (optional) $NXmanufacturer$

**ion_gun**: (optional) $NXsource$

The source which creates the ion beam.

**name**: (optional) $NX_CHAR$

Given name/alias for the ion gun.

**emitter_type**: (optional) $NX_CHAR$

Emitter type used to create the ion beam.

If the emitter type is other, give further details in the description field.

Any of these values: liquid_metal | plasma | gas_field | other

**description**: (optional) $NX_CHAR$

Ideally, a (globally) unique persistent identifier, link, or text to a resource which gives further details.

**brightness**: (optional) $NX_FLOAT$ \{units=$NX_ANY$\}

Average/nominal brightness

**current**: (optional) $NX_FLOAT$ \{units=$NX_CURRENT$\}

Charge current

**voltage**: (optional) $NX_FLOAT$ \{units=$NX_VOLTAGE$\}

Ion acceleration voltage upon source exit and entering the vacuum flight path.

**ion_energy_profile**: (optional) $NX_NUMBER$ \{units=$NX_ENERGY$\}

**probe**: (optional) $NXion$

Which ionized elements or molecular ions form the beam. Examples are gallium, helium, neon, argon, krypton, or xenon, O2+.

**TRANSFORMATIONS**: (optional) $NXtransformations$

Affine transformation which detail the arrangement in the microscope relative to the optical axis and beam path.

**APERTURE_EM**: (optional) $NXaperture_em$

**LENS_EM**: (optional) $NXlens_em$

**SENSOR**: (optional) $NXsensor$

**BEAM**: (optional) $NXbeam$

Individual characterization results for the position, shape, and characteristics of the ion beam.

NXtransformations should be used to specify the location or position at which details about the ion beam are probed.
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXibeam_column/APERTURE_EM-group
- /NXibeam_column/BEAM-group
- /NXibeam_column/ion_gun-group
- /NXibeam_column/ion_gun/brightness-field
- /NXibeam_column/ion_gun/current-field
- /NXibeam_column/ion_gun/description-field
- /NXibeam_column/ion_gun/emitter_type-field
- /NXibeam_column/ion_gun/ion_energy_profile-field
- /NXibeam_column/ion_gun/name-field
- /NXibeam_column/ion_gun/probe-group
- /NXibeam_column/ion_gun/TRANSFORMATIONS-group
- /NXibeam_column/ion_gun/voltage-field
- /NXibeam_column/LENS_EM-group
- /NXibeam_column/MANUFACTURER-group
- /NXibeam_column/SENSOR-group

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXibeam_column.nxdl.xml

NXimage_set_em_adf

Status:
base class, extends NXobject

Description:
Container for reporting a set of annular dark field images.

Symbols:
- n_images: Number of images
- n_y: Number of pixel per image in the slow direction
- n_x: Number of pixel per image in the fast direction

Groups cited:
NXdata, NXprocess

Structure:

PROCESS: (optional) NXprocess
Details about how the images were processed from the detector readings.

program: (optional) NX_CHAR
Commercial or otherwise given name to the program which was used to process detector data into the adf image(s).

@version: (optional) NX_CHAR

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

adf_inner_half_angle: (optional) NX_FLOAT \{units=NX_ANGLE\}

Annulus inner half angle

adf_outer_half_angle: (optional) NX_FLOAT \{units=NX_ANGLE\}

Annulus outer half angle

DATA: (optional) NXdata

Annular dark field images.

@long_name: (optional) NX_CHAR

Image intensities

intensity: (optional) NX_NUMBER (Rank: 3, Dimensions: [n_images, n_y, n_x]) \{units=NX_UNITLESS\}

Image intensity values.

image_id: (optional) NX_UINT (Rank: 1, Dimensions: [n_images]) \{units=NX_UNITLESS\}

Image identifier

@long_name: (optional) NX_CHAR

Image ID.

ypos: (optional) NX_NUMBER (Rank: 1, Dimensions: [n_y]) \{units=NX_LENGTH\}

Pixel center of mass position y-coordinates.

@long_name: (optional) NX_CHAR

Label for the y axis.

xpos: (optional) NX_NUMBER (Rank: 1, Dimensions: [n_x]) \{units=NX_LENGTH\}

Pixel center of mass position x-coordinates.

@long_name: (optional) NX_CHAR

Label for the x axis.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXimage_set_em_adf/DATA-group
- /NXimage_set_em_adf/DATA/image_id-field
- /NXimage_set_em_adf/DATA/image_id@long_name-attribute
- /NXimage_set_em_adf/DATA/intensity-field
• /NXimage_set_em_adf/DATA/xpos-field
• /NXimage_set_em_adf/DATA/xpos@long_name-attribute
• /NXimage_set_em_adf/DATA/ypos-field
• /NXimage_set_em_adf/DATA/ypos@long_name-attribute
• /NXimage_set_em_adf/DATA@long_name-attribute
• /NXimage_set_em_adf/PROCESS-group
• /NXimage_set_em_adf/PROCESS/adf_inner_half_angle-field
• /NXimage_set_em_adf/PROCESS/adf_outer_half_angle-field
• /NXimage_set_em_adf/PROCESS/program-field
• /NXimage_set_em_adf/PROCESS/program@version-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXimage_set_em_adf.nxdl.xml

NXimage_set_em_bf

Status:
base class, extends NXobject

Description:
Container for reporting a set of images taken in bright field mode.

Symbols:
No symbol table

Groups cited:
NXdata, NXprocess

Structure:
DATA: (optional) NXdata
PROCESS: (optional) NXprocess

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

• /NXimage_set_em_bf/DATA-group
• /NXimage_set_em_bf/PROCESS-group

NXDL Source:
NXimage_set_em_bse

Status:
base class, extends NXobject

Description:
Container for reporting a set of back-scattered electron images.

Symbols:
No symbol table

Groups cited:
NXdata, NXprocess

Structure:
DATA: (optional) NXdata
PROCESS: (optional) NXprocess

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXimage_set_em_bse/DATA-group
- /NXimage_set_em_bse/PROCESS-group

NXDL Source:

NXimage_set_em_chamber

Status:
base class, extends NXobject

Description:
Container for images recorded with e.g. a TV camera in the microscope chamber.

Symbols:
No symbol table

Groups cited:
NXdata, NXprocess

Structure:
DATA: (optional) NXdata
PROCESS: (optional) NXprocess
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXimage_set_em_chamber/DATA-group
- /NXimage_set_em_chamber/PROCESS-group

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXimage_set_em_chamber.nxdl.xml

NXimage_set_em_df

Status:
base class, extends NXobject

Description:
Container for reporting a set of images taken in dark field mode.

Symbols:
No symbol table

Groups cited:
NXdata, NXprocess

Structure:

DATA: (optional) NXdata

PROCESS: (optional) NXprocess

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXimage_set_em_df/DATA-group
- /NXimage_set_em_df/PROCESS-group

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXimage_set_em_df.nxdl.xml

NXimage_set_em_diffrac

Status:
base class, extends NXobject

Description:
Container for reporting a set of diffraction images.

Symbols:
No symbol table

Groups cited:
NXdata, NXprocess
Structure:

- DATA: (optional) NXdata
- PROCESS: (optional) NXprocess

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXimage_set_em_diffrac/DATA-group
- /NXimage_set_em_diffrac/PROCESS-group

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXimage_set_em_diffrac.nxdl.xml

NXimage_set_em_ecci

Status:
base class, extends NXobject

Description:
Container for reporting back-scattered electron channeling contrast images.

Symbols:
No symbol table

Groups cited:
NXdata, NXprocess

Structure:

- DATA: (optional) NXdata
- PROCESS: (optional) NXprocess

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXimage_set_em_ecci/DATA-group
- /NXimage_set_em_ecci/PROCESS-group

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXimage_set_em_ecci.nxdl.xml
NXimage_set_em_kikuchi

Status:

base class, extends NXobject

Description:

Electron backscatter diffraction (EBSD) Kikuchi pattern.

The container can also store data related to a post-processing of these Kikuchi pattern, which is the backbone of orientation microscopy especially in materials science and materials engineering.

Based on a fuse of the M. A. Jackson et al. of the DREAM.3D community and the open H5OINA format of Oxford Instruments P. Pinard et al.

EBSD can be used, usually with FIB/SEM microscopes, for three-dimensional orientation microscopy. So-called serial section analyses. For a detailed overview of these techniques see e.g.

- M. A. Groeber et al.
- A. J. Schwartz et al.
- P. A. Rottman et al.

With serial-sectioning this involves however always a sequence of measuring, milling. In this regard, each serial section (measuring) and milling is an own NXevent_data_em instance and thus there such a three-dimensional characterization should be stored as a set of two-dimensional data, with as many NXevent_data_em instances as sections were measured.

These measured serial sectioning images need virtually always post-processing to arrive at the aligned and cleaned image stack respective digital microstructure representation as (a representative) volume element. Several software packages are available for this post-processing. For now we do not consider metadata of these post-processing steps as a part of this base class.

Symbols:

- \( n_p \): Number of scan points, one pattern per scan point.
- \( n_y \): Number of pixel per Kikuchi pattern in the slow direction
- \( n_x \): Number of pixel per Kikuchi pattern in the fast direction

Groups cited:

NXcollection, NXdata, NXprocess

Structure:

- grid_type: (optional) NX_CHAR
  Which pixel primitive shape is used.
  Any of these values: square | hexagon

- step_size: (optional) NX_NUMBER (Rank: 1, Dimensions: [2]) \{units=NX_LENGTH\}
  The prescribed step size. First value is for the slow changing, second value is for the fast changing dimension.

DATA: (optional) NXdata

Collected Kikuchi pattern as an image stack.

- intensity: (optional) NX_NUMBER (Rank: 3, Dimensions: \([n_p, n_y, n_x]\) \{units=NX_UNITLESS\}

@long_name: (optional) NX_CHAR
Kikuchi pattern intensity

**image_id**: (optional)  
NX_NUMBER (Rank: 1, Dimensions: [n_p]) \{units=NX_UNITLESS\}

@long_name: (optional)  
NX_CHAR

Kikuchi pattern identifier

**ypos**: (optional)  
NX_NUMBER (Rank: 1, Dimensions: [n_y]) \{units=NX_LENGTH\}

@long_name: (optional)  
NX_CHAR

Label for the y axis

**xpos**: (optional)  
NX_NUMBER (Rank: 1, Dimensions: [n_x]) \{units=NX_LENGTH\}

@long_name: (optional)  
NX_CHAR

Label for the x axis

**calibration**: (optional)  
NXprocess

OIM, orientation imaging microscopy. Post-processing of the Kikuchi pattern to identify orientations.

**pattern_quality**: (optional)  
NX_FLOAT (Rank: 1, Dimensions: [n_p]) \{units=NX_UNITLESS\}

**pattern_center**: (optional)  
NX_NUMBER (Rank: 1, Dimensions: [n_p]) \{units=NX_LENGTH\}

**detector_distance**: (optional)  
NX_FLOAT (Rank: 1, Dimensions: [n_p]) \{units=NX_LENGTH\}

**background_correction**: (optional)  
NXprocess

Details about the background correction applied to each Kikuchi pattern.

**band_detection**: (optional)  
NXprocess

**mode**: (optional)  
NX_CHAR

How are Kikuchi bands detected

Obligatory value: center

**band_contrast**: (optional)  
NX_NUMBER (Rank: 1, Dimensions: [n_p]) \{units=NX_UNITLESS\}

**band_slope**: (optional)  
NX_NUMBER (Rank: 1, Dimensions: [n_p]) \{units=NX_UNITLESS\}

**bands**: (optional)  
NX_NUMBER (Rank: 1, Dimensions: [n_p]) \{units=NX_UNITLESS\}

How many bands were detected in the pattern.

**indexing**: (optional)  
NXprocess

**mode**: (optional)  
NX_CHAR

How are pattern being indexed?

Obligatory value: optimize_bd

**min_bands**: (optional)  
NX_NUMBER (Rank: 1, Dimensions: [n_p])
Minimum number of bands required to index the pattern

**status**: (optional) `NX_NUMBER` (Rank: 1, Dimensions: [n_p])

{units=`NX_UNITLESS`}

Which return value did the indexing algorithm yield for each pattern.

- Details about bad pixels
- Too high angular deviation
- No solution
- Not analyzed
- Success
- Unexpected errors

**phase_identifier**: (optional) `NX_UINT` (Rank: 1, Dimensions: [n_p])

{units=`NX_UNITLESS`}

Labels referring to the phase_identifier for each pattern (from reflectors) that matched best.

**mean_angular_deviation**: (optional) `NX_FLOAT` (Rank: 1, Dimensions: [n_p])

{units=`NX_ANGLE`}

**confidence_index**: (optional) `NX_FLOAT` (Rank: 1, Dimensions: [n_p])

{units=`NX_UNITLESS`}

**reflector**: (optional) `NXcollection`

Lattice planes used as reflectors for indexing pattern in electron-backscatter diffraction (EBSD). One collection for each reflector.

**unit_cell_abc**: (optional) `NX_FLOAT` (Rank: 1, Dimensions: [3])

{units=`NX_LENGTH`}

Crystallography unit cell parameters a, b, and c

**unit_cell_alphabetagamma**: (optional) `NX_FLOAT` (Rank: 1, Dimensions: [3])

{units=`NX_ANGLE`}

Crystallography unit cell parameters alpha, beta, and gamma

**unit_cell_class**: (optional) `NX_CHAR`

Any of these values:

- triclinic
- monoclinic
- orthorhombic
- tetragonal
- rhombohedral
- hexagonal
- cubic

**space_group**: (optional) `NX_CHAR`

Crystallographic space group

**laue_group**: (optional) `NX_CHAR`
Laue group

**phase_identifier**: (optional) *NX_UINT* \{units=*NX_UNITLESS*\}

Numeral identifier for each phase. The value 0 is reversed for the unknown phase.

**phase_name**: (optional) *NX_CHAR*

Name of the phase/alias.

**number_of_reflectors**: (optional) *NX_UINT*

**miller_indices**: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [number_of_reflectors, 6]) \{units=*NX_UNITLESS*\}

Miller indices \((hkl)\)\([uvw]\).

**binning**: (optional) *NXcollection*

**mode**: (optional) *NX_CHAR*

Free-text description for instrument specific settings.

**binning**: (optional) *NX_UINT* (Rank: 1, Dimensions: [2]) \{units=*NX_UNITLESS*\}

How is the camera signal binned. First the number of pixel along the slow direction. Second the number of pixel along the fast direction.

**hough_transformation**: (optional) *NXprocess*

**resolution**: (optional) *NX_NUMBER* \{units=*NX_UNITLESS*\}

**profiling**: (optional) *NXcollection*

**acquisition_speed**: (optional) *NX_FLOAT* \{units=*NX_FREQUENCY*\}

Average number of patterns taken on average.

**acquisition_time**: (optional) *NX_FLOAT* \{units=*NX_TIME*\}

Wall-clock time the acquisition took.

**hit_rate**: (optional) *NX_FLOAT* \{units=*NX_DIMENSIONLESS*\}

Fraction of successfully indexed pattern of the set.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXimage_set_em_kikuchi/binning-group
- /NXimage_set_em_kikuchi/binning/binning-field
- /NXimage_set_em_kikuchi/binning/mode-field
- /NXimage_set_em_kikuchi/calibration-group
- /NXimage_set_em_kikuchi/DATA-group
- /NXimage_set_em_kikuchi/DATA/image_id-field
- /NXimage_set_em_kikuchi/DATA/image_id@long_name-attribute
- /NXimage_set_em_kikuchi/DATA/intensity-field
- /NXimage_set_em_kikuchi/DATA/intensity@long_name-attribute
3.3. NeXus Class Definitions

- /NXimage_set_em_kikuchi/DATA/xpos-field
- /NXimage_set_em_kikuchi/DATA/xpos@long_name-attribute
- /NXimage_set_em_kikuchi/DATA/ypos-field
- /NXimage_set_em_kikuchi/DATA/ypos@long_name-attribute
- /NXimage_set_em_kikuchi/grid_type-field
- /NXimage_set_em_kikuchi/hough_transformation-group
  - /NXimage_set_em_kikuchi/hough_transformation/resolution-field
- /NXimage_set_em_kikuchi/oim-group
  - /NXimage_set_em_kikuchi/oim/background_correction-group
  - /NXimage_set_em_kikuchi/oim/band_detection-group
    - /NXimage_set_em_kikuchi/oim/band_detection/band_contrast-field
    - /NXimage_set_em_kikuchi/oim/band_detection/band_slope-field
    - /NXimage_set_em_kikuchi/oim/band_detection/bands-field
  - /NXimage_set_em_kikuchi/oim/detector_distance-field
  - /NXimage_set_em_kikuchi/oim/indexing-group
    - /NXimage_set_em_kikuchi/oim/indexing/confidence_index-field
    - /NXimage_set_em_kikuchi/oim/indexing/mean_angular_deviation-field
    - /NXimage_set_em_kikuchi/oim/indexing/min_bands-field
    - /NXimage_set_em_kikuchi/oim/indexing/mode-field
    - /NXimage_set_em_kikuchi/oim/indexing/phase_identifier-field
  - /NXimage_set_em_kikuchi/oim/indexing/reflector-group
    - /NXimage_set_em_kikuchi/oim/indexing/reflector/laue_group-field
    - /NXimage_set_em_kikuchi/oim/indexing/reflector/miller_indices-field
    - /NXimage_set_em_kikuchi/oim/indexing/reflector/number_of_reflectors-field
    - /NXimage_set_em_kikuchi/oim/indexing/reflector/phase_identifier-field
    - /NXimage_set_em_kikuchi/oim/indexing/reflector/phase_name-field
    - /NXimage_set_em_kikuchi/oim/indexing/reflector/space_group-field
    - /NXimage_set_em_kikuchi/oim/indexing/reflector/unit_cell_abc-field
    - /NXimage_set_em_kikuchi/oim/indexing/reflector/unit_cell_alphabetagamma-field
    - /NXimage_set_em_kikuchi/oim/indexing/reflector/unit_cell_class-field
    - /NXimage_set_em_kikuchi/oim/indexing/status-field
  - /NXimage_set_em_kikuchi/oim/pattern_center-field
  - /NXimage_set_em_kikuchi/oim/pattern_quality-field
- /NXimage_set_em_kikuchi/profiling-group
  - /NXimage_set_em_kikuchi/profiling/acquisition_speed-field
NXimage_set_em_kikuchi

bullet /NXimage_set_em_kikuchi/profiling/acquisition_time-field
bullet /NXimage_set_em_kikuchi/profiling/hit_rate-field
bullet /NXimage_set_em_kikuchi/step_size-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXimage_set_em_kikuchi.nxdl.xml

NXimage_set_em_ronchigram

Status:
base class, extends NXobject

Description:
Container for reporting a set of images related to a ronchigram.
Ronchigrams are specifically used in transmission electron microscopy to judge the settings for the aberration corrections and alignment.

Symbols:
No symbol table

Groups cited:
NXdata, NXprocess

Structure:
DATA: (optional) NXdata
PROCESS: (optional) NXprocess

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.
bullet /NXimage_set_em_ronchigram/DATA-group
bullet /NXimage_set_em_ronchigram/PROCESS-group

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXimage_set_em_ronchigram.nxdl.xml

NXimage_set_em_se

Status:
base class, extends NXobject

Description:
Container for reporting a set of secondary electron images.
Secondary electron images are one of the most important signal especially for scanning electron microscopy in materials science and engineering, for analyses of surface topography, getting an overview of the analysis region, or fractography.
Symbols:

- \( n_{\text{images}} \): Number of images.
- \( n_y \): Number of pixels along the slow scan direction (often y).
- \( n_x \): Number of pixels along the fast scan direction (often x).

Groups cited:

- NXdata, NXoptical_system_em, NXprocess

Structure:

- \( \text{roi} \): (optional) \( \text{NX\_NUMBER} \) (Rank: 2, Dimensions: \([n_{\text{images}}, 4]\)) \{units=NX\_LENGTH\}
  
  Physical dimensions of the region-of-interest on the specimen surface which the image covers. The first and second number are the coordinates for the minimum edge, the third and fourth number are the coordinates for the maximum edge of the rectangular ROI.

- \( \text{dwell\_time} \): (optional) \( \text{NX\_FLOAT} \) \{units=NX\_TIME\}

- \( \text{number\_of\_frames\_averaged} \): (optional) \( \text{NX\_UINT} \) (Rank: 1, Dimensions: \([n_{\text{images}}]\)) \{units=NX\_UNITLESS\}
  
  How many frames were averaged to obtain the image.

- \( \text{bias\_voltage} \): (optional) \( \text{NX\_FLOAT} \)

  Bias voltage applied to the e.g. Faraday cage in front of the SE detector to attract or repel secondary electrons below a specific kinetic energy.

DATA: (optional) NXdata

Collected secondary electron images (eventually as an image stack).

- \( \text{intensity} \): (optional) \( \text{NX\_NUMBER} \) (Rank: 3, Dimensions: \([n_p, n_y, n_x]\)) \{units=NX\_UNITLESS\}
  
  @long\_name: (optional) \( \text{NX\_CHAR} \)

  Secondary electron image intensity

- \( \text{image\_id} \): (optional) \( \text{NX\_NUMBER} \) (Rank: 1, Dimensions: \([n_p]\)) \{units=NX\_UNITLESS\}
  
  @long\_name: (optional) \( \text{NX\_CHAR} \)

  Image identifier

- \( \text{ypos} \): (optional) \( \text{NX\_NUMBER} \) (Rank: 1, Dimensions: \([n_y]\)) \{units=NX\_LENGTH\}
  
  @long\_name: (optional) \( \text{NX\_CHAR} \)

  Label for the y axis

- \( \text{xpos} \): (optional) \( \text{NX\_NUMBER} \) (Rank: 1, Dimensions: \([n_x]\)) \{units=NX\_LENGTH\}
  
  @long\_name: (optional) \( \text{NX\_CHAR} \)

  Label for the x axis

OPTICAL\_SYSTEM\_EM: (optional) NXoptical_system_em

Container to store relevant settings affecting beam path, magnification, and working_distance

- \( \text{scan\_rotation} \): (optional) NXprocess

  Scan rotation is an option offer by most control software.

- \( \text{tilt\_correction} \): (optional) NXprocess
Tilt correction is an option offered by most control software of SEMs to apply an on-the-fly processing of the image to correct for the excursion of objects when characterized with SE images on samples whose surface normal is tilted about an axis perpendicular to the optical axis.

**active**: (optional) **NX_BOOLEAN**

Was the option switched active or not.

**dynamic_focus**: (optional) **NXprocess**

Dynamic focus is an option offered by most control software of SEMs to keep the image also focused when probing locations on the specimens beyond those for which the lens system was calibrated.

**active**: (optional) **NX_BOOLEAN**

Was the option switched active or not.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXimage_set_em_se/bias_voltage-field
- /NXimage_set_em_se/DATA-group
- /NXimage_set_em_se/DATA/image_id-field
- /NXimage_set_em_se/DATA/image_id@long_name-attribute
- /NXimage_set_em_se/DATA/intensity-field
- /NXimage_set_em_se/DATA/intensity@long_name-attribute
- /NXimage_set_em_se/DATA/xpos-field
- /NXimage_set_em_se/DATA/xpos@long_name-attribute
- /NXimage_set_em_se/DATA/ypos-field
- /NXimage_set_em_se/DATA/ypos@long_name-attribute
- /NXimage_set_em_se/dwell_time-field
- /NXimage_set_em_se/dynamic_focus-group
- /NXimage_set_em_se/dynamic_focus/active-field
- /NXimage_set_em_se/number_of_frames_averaged-field
- /NXimage_set_em_se/OPTICAL_SYSTEM_EM-group
- /NXimage_set_em_se/roi-field
- /NXimage_set_em_se/scan_rotation-group
- /NXimage_set_em_se/tilt_correction-group
- /NXimage_set_em_se/tilt_correction/active-field

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXimage_set_em_se.nxdl.xml
NXinteraction_vol_em

Status:

base class, extends NXobject

Description:

Base class for storing details about a modelled shape of interaction volume.

The interaction volume is mainly relevant in scanning electron microscopy when the sample is thick enough so that the beam is unable to illuminate through the specimen. Computer models like Monte Carlo or molecular dynamics / electron beam interaction simulations can be used to qualify and/or quantify the shape of the interaction volume.

Explicit or implicit descriptions are possible.

• An implicit description is via a set of electron/specimen interactions represented ideally as trajectory data from the computer simulation.

• An explicit description is via an iso-contour surface using either a simulation grid or a triangulated surface mesh of the approximated iso-contour surface evaluated at specific threshold values. Iso-contours could be computed from electron or particle fluxes through an imaginary control surface (the iso-surface). Threshold values can be defined by particles passing through a unit control volume (electrons) or energy-levels (e.g. the case of X-rays). Details depend on the model.

• Another explicit description is via theoretical models which may be relevant e.g. for X-ray spectroscopy

Further details on how the interaction volume can be quantified is available in the literature for example:

• S. Richter et al.
• J. Bünger et al.

Symbols:

No symbol table

Groups cited:

NXdata, NXprocess

Structure:

DATA: (optional) NXdata

PROCESS: (optional) NXprocess

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

• /NXinteraction_vol_em/DATA-group
• /NXinteraction_vol_em/PROCESS-group

NXDL Source:
**NXion**

**Status:**
- base class, extends *NXobject*

**Description:**
- Set of atoms of a molecular ion or fragment in e.g. ToF mass spectrometry.

**Symbols:**
- The symbols used in the schema to specify e.g. dimensions of arrays.
  - **n_ivecmax**: Maximum number of atoms/isotopes allowed per (molecular) ion (fragment).
  - **n_ranges**: Number of mass-to-charge-state-ratio range intervals for ion type.

**Groups cited:**
- none

**Structure:**

- **ion_type**: (optional) *NX_UINT* {units=*NX_UNITLESS*}
  - Ion type (ion species) identifier. The identifier zero is reserved for the special unknown ion type.

- **isotope_vector**: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_ivecmax]) {units=*NX_UNITLESS*}
  - A vector of isotope hash values. These values have to be stored in an array, sorted in decreasing order. The array is filled with zero hash values indicating unused places. The individual hash values are built with the following hash function:
    
    \[ H = Z + N \times 256 \]
    
    with \( Z \) the number of protons and \( N \) the number of neutrons of each isotope respectively.
    
    \( Z \) and \( N \) have to be 8-bit unsigned integers. For the rationale behind this M. Kühbach et al. (2021)

- **charge_state**: (optional) *NX_INT* {units=*NX_DIMENSIONLESS*}
  - Signed charge state of the ion in multiples of electron charge.
  - Only positive values will be measured in atom probe microscopy as the ions are accelerated by a negatively signed bias electric field. In the case that the charge state is not explicitly recoverable, the value should be set to zero.
  - In atom probe microscopy this is for example the case when using classical range file formats like RNG, RRNG for atom probe data. These file formats do not document the charge state explicitly. They report the number of atoms of each element per molecular ion surplus the mass-to-charge-state-ratio interval. With this it is possible to recover the charge state only for specific molecular ions as the accumulated mass of the molecular ion is defined by the isotopes, which without knowing the charge leads to an underconstrained problem. Details on ranging can be found in the literature: M. K. Miller

- **name**: (optional) *NX_CHAR*
  - Human-readable ion type name (e.g. Al+++) The string should consists of ASCII UTF-8 characters, ideally using LaTeX notation to specify the isotopes, ions, and charge state. Examples are 12C + or Al +++. Although this name may be human-readable and intuitive, parsing such names becomes impractical for more complicated cases. Therefore, the isotope_vector should be the preferred machine-readable format to use.

- **mass_to_charge_range**: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [n_ranges, 2]) {units=*NX_ANY*}
Associated lower \((mq_{\text{min}})\) and upper \((mq_{\text{max}})\) bounds of mass-to-charge-state ratio interval(s) \([mq_{\text{min}}, mq_{\text{max}}]\) (boundaries included) for which the respective ion is labelled as an ion of the here referred to \textit{ion\_type}.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- \textit{/NXion/charge\_state-field}
- \textit{/NXion/ion\_type-field}
- \textit{/NXion/isotope\_vector-field}
- \textit{/NXion/mass\_to\_charge\_range-field}
- \textit{/NXion/name-field}

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXion.nxdl.xml

**NXiens\_em**

**Status:**

base class, extends \textit{NXobject}

**Description:**

Description of an electro-magnetic lens or a compound lens.

For \textit{NXtransformation}s the origin of the coordinate system is placed in the center of the lens (its polepiece, pinhole, or another point of reference). The origin should be specified in the \textit{NXtransformation}s.

For details of electro-magnetic lenses in the literature see e.g. L. Reimer

**Symbols:**

No symbol table

**Groups cited:**

\textit{NXmanufacturer}, \textit{NXtransformation}s

**Structure:**

- \textit{type}: (optional) \textit{NX\_CHAR}
  
  Qualitative type of lens with respect to the number of pole pieces.

  Any of these values:
  
  - \textit{single}
  - \textit{double}
  - \textit{quadrupole}
  - \textit{hexapole}
  - \textit{octupole}

- \textit{name}: (optional) \textit{NX\_CHAR}
  
  Given name, alias, colloquial, or short name for the lens. For manufacturer names and identifiers use respective manufacturer fields.
**manufacturer_name**: (optional) *NX_CHAR*

Name of the manufacturer who built/constructed the lens.

**model**: (optional) *NX_CHAR*

Hardware name, hash identifier, or serial number that was given by the manufacturer to identify the lens.

**description**: (optional) *NX_CHAR*

Ideally an identifier, persistent link, or free text which gives further details about the lens.

**voltage**: (optional) *NX_NUMBER* {units=*NX_VOLTAGE*}

Excitation voltage of the lens. For dipoles it is a single number. For higher orders, it is an array.

**current**: (optional) *NX_NUMBER* {units=*NX_CURRENT*}

Excitation current of the lens. For dipoles it is a single number. For higher orders, it is an array.

**depends_on**: (optional) *NX_CHAR*

Specifies the position of the lens by pointing to the last transformation in the transformation chain in the NXtransformations group.

**MANUFACTURER**: (optional) *NXmanufacturer*

**TRANSFORMATIONS**: (optional) *NXtransformations*

Collection of axis-based translations and rotations to describe the location and geometry of the lens as a component in the instrument. Typically, the components of a system should all be related relative to each other and only one component should relate to the reference coordinate system.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXlens_em/current-field
- /NXlens_em/depends_on-field
- /NXlens_em/description-field
- /NXlens_em/MANUFACTURER-group
- /NXlens_em/manufacturer_name-field
- /NXlens_em/model-field
- /NXlens_em/name-field
- /NXlens_em/TRANSFORMATIONS-group
- /NXlens_em/type-field
- /NXlens_em/voltage-field

**NXDL Source:**

**NXmagnetic_kicker**

**Status:**
base class, extends *NXobject*

**Description:**
definition for a magnetic kicker.

**Symbols:**
No symbol table

**Groups cited:**
*NXlog*

**Structure:**

- **description**: (optional) *NX_CHAR*
  extended description of the kicker.

- **beamline_distance**: (optional) *NX_FLOAT* {units=*NX_LENGTH*}
  define position of beamline element relative to production target

- **timing**: (optional) *NX_FLOAT* {units=*NX_TIME*}
  kicker timing as defined by `description` attribute

  - **@description**: (optional) *NX_CHAR*

- **set_current**: (optional) *NX_FLOAT* {units=*NX_CURRENT*}
  current set on supply.

- **set_voltage**: (optional) *NX_FLOAT* {units=*NX_VOLTAGE*}
  voltage set on supply.

- **read_current**: (optional) *NXlog*
  current read from supply.

  - **value**: (optional) *NX_CHAR* {units=*NX_CURRENT*}

- **read_voltage**: (optional) *NXlog*
  voltage read from supply.

  - **value**: (optional) *NX_CHAR* {units=*NX_VOLTAGE*}

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXmagnetic_kicker/beamline_distance-field`
- `/NXmagnetic_kicker/description-field`
- `/NXmagnetic_kicker/read_current-group`
- `/NXmagnetic_kicker/read_current/value-field`
- `/NXmagnetic_kicker/read_voltage-group`
- `/NXmagnetic_kicker/read_voltage/value-field`
• /NXmagnetic_kicker/set_current-field
• /NXmagnetic_kicker/set_voltage-field
• /NXmagnetic_kicker/timing-field
• /NXmagnetic_kicker/timing@description-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmagnetic_kicker.nxdl.xml

NXmanipulator

Status:
base class, extends NXobject

Description:
Extension of NXpositioner to include fields to describe the use of manipulators in photoemission experiments.

Symbols:
No symbol table

Groups cited:
NXpositioner, NXtransformations

Structure:

name: (optional) NX_CHAR
Name of the manipulator.

description: (optional) NX_CHAR
A description of the manipulator.

type: (optional) NX_CHAR
Type of manipulator, Hexapod, Rod, etc.

cryocoolant: (optional) NX_BOOLEAN
Is cryocoolant flowing through the manipulator?

cryostat_temperature: (optional) NX_FLOAT {units=NX_TEMPERATURE}
Temperature of the cryostat (coldest point)

heater_power: (optional) NX_FLOAT {units=NX_POWER}
Power in the heater for temperature control.

sample_temperature: (optional) NX_FLOAT {units=NX_TEMPERATURE}
Temperature at the closest point to the sample. This field may also be found in NXsample if present.

drain_current: (optional) NX_FLOAT {units=NX_CURRENT}
Current to neutralize the photoemission current. This field may also be found in NXsample if present.

sample_bias: (optional) NX_FLOAT {units=NX_CURRENT}
Possible bias of the sample with respect to analyser ground. This field may also be found in NXSample if present.

**depends_on:** (optional) *NX_CHAR*

Refers to the last transformation specifying the position of the manipulator in the NXtransformations chain.

**POSITIONER:** (optional) *NXpositioner*

Class to describe the motors that are used in the manipulator

**TRANSFORMATIONS:** (optional) *NXtransformations*

Collection of axis-based translations and rotations to describe the location and geometry of the manipulator as a component in the instrument. Conventions from the NXtransformations base class are used. In principle, the McStas coordinate system is used. The first transformation has to point either to another component of the system or . (for pointing to the reference frame) to relate it relative to the experimental setup. Typically, the components of a system should all be related relative to each other and only one component should relate to the reference coordinate system.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXmanipulator/cryocoolant-field
- /NXmanipulator/cryostat_temperature-field
- /NXmanipulator/depends_on-field
- /NXmanipulator/description-field
- /NXmanipulator/drain_current-field
- /NXmanipulator/heater_power-field
- /NXmanipulator/name-field
- /NXmanipulator/POSITIONER-group
- /NXmanipulator/sample_bias-field
- /NXmanipulator/sample_temperature-field
- /NXmanipulator/TRANSFORMATIONS-group
- /NXmanipulator/type-field

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmanipulator.nxdl.xml
**NXmanufacturer**

**Status:**
base class, extends *NXobject*

**Description:**
Details about a component as defined by its manufacturer.

**Symbols:**
No symbol table

**Groups cited:**
none

**Structure:**
- **name:** (optional) *NX_CHAR*
  Company name of the manufacturer.
- **model:** (optional) *NX_CHAR*
  Version or model of the component named by the manufacturer.
- **identifier:** (optional) *NX_CHAR*
  Ideally, (globally) unique persistent identifier, i.e. a serial number or hash identifier of the component.
- **capability:** (optional) *NX_CHAR*
  Free-text list with eventually multiple terms of functionalities which the component offers.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXmanufacturer/capability-field`
- `/NXmanufacturer/identifier-field`
- `/NXmanufacturer/model-field`
- `/NXmanufacturer/name-field`

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmanufacturer.nxdl.xml

**NXmpes**

**Status:**
application definition, extends *NXobject*

**Description:**
This is the most general application definition for multidimensional photoelectron spectroscopy.

**Symbols:**
No symbol table
Groups cited:
  NXaperture, NXbeam, NXcalibration, NXcollectioncolumn, NXdata, NXdetector, NXelectronanalyser, NXenergydispersion, NXentry, NXinstrument, NXmanipulator, NXnote, NXprocess, NXsample, NXsource

Structure:

ENTRY: (required) NXentry
  title: (required) NX_CHAR
  start_time: (required) NX_DATE_TIME
    Datetime of the start of the measurement.
  definition: (required) NX_CHAR
    Obligatory value: NXmpes
  @version: (required) NX_CHAR

INSTRUMENT: (required) NXinstrument
  energy_resolution: (required) NX_FLOAT

SOURCE: (required) NXsource
  The source used to generate the primary photons. Properties refer strictly to parameters of the source, not of the output beam. For example, the energy of the source is not the optical power of the beam, but the energy of the electron beam in a synchrotron and so on.
  type: (required) NX_CHAR
    Any of these values:
    • Synchrotron X-ray Source
    • Rotating Anode X-ray
    • Fixed Tube X-ray
    • UV Laser
    • Free-Electron Laser
    • Optical Laser
    • UV Plasma Source
    • Metal Jet X-ray
    • HHG laser
  name: (required) NX_CHAR
  probe: (required) NX_CHAR
    Type of probe. In photoemission it's always photons, so the full NIAC list is restricted.
    Any of these values: x-ray | ultraviolet | visible light

BEAM: (required) NXbeam
  distance: (required) NX_NUMBER
    Distance of the point of evaluation of the beam from the sample surface.
incident_energy: (required) `NX_FLOAT`
incident_energy_spread: (recommended) `NX_NUMBER`
incident_polarization: (recommended) `NX_NUMBER`

**ELECTRONANALYSER:** (required) `NXelectronanalyser`

description: (required) `NX_CHAR`

energy_resolution: (recommended) `NX_FLOAT`

Energy resolution of the analyser with the current setting. May be linked from a NXcalibration.

fast_axes: (recommended) `NX_CHAR`
slow_axes: (recommended) `NX_CHAR`

**COLLECTIONCOLUMN:** (required) `NXcollectioncolumn`
scheme: (required) `NX_CHAR`

Scheme of the electron collection column.

Any of these values:

- Standard
- Angular dispersive
- Selective area
- Deflector
- PEEM
- Momentum Microscope

mode: (recommended) `NX_CHAR`

projection: (recommended) `NX_CHAR`

field_aperture: (optional) `NXaperture`

The size and position of the field aperture inserted in the column. To add additional or other apertures use the APERTURE group of NXcollectioncolumn.

contrast_aperture: (optional) `NXaperture`

The size and position of the contrast aperture inserted in the column. To add additional or other apertures use the APERTURE group of NXcollectioncolumn.

**ENERGYDISPERSION:** (required) `NXenergydispersion`
scheme: (required) `NX_CHAR`

Any of these values:

- tof
- hemispherical
- double hemispherical
- cylindrical mirror
- display mirror
• retarding grid

pass_energy: (required) NX_FLOAT

detector_type: (required) NX_CHAR

energy_scan_mode: (required) NX_CHAR

entrance_slit: (optional) NXAperature

Size, position and shape of the entrance slit in dispersive analyzers. To add additional or other slits use the APERTURE group of NXenergydispersion.

exit_slit: (optional) NXAperature

Size, position and shape of the exit slit in dispersive analyzers. To add additional or other slits use the APERTURE group of NXenergydispersion.

DETECTOR: (required) NXdetector

amplifier_type: (recommended) NX_CHAR

Type of electron amplifier in the first amplification step.

Any of these values: MCP | channeltron

detector_type: (recommended) NX_CHAR

Description of the detector type.

Any of these values:

• DLD
• Phosphor+CCD
• Phosphor+CMOS
• ECMOS
• Anode
• Multi-anode

DATA: (recommended) NXdata

@signal: (required) NX_CHAR

Obligatory value: raw

raw: (required) NX_NUMBER

Raw data before calibration.

MANIPULATOR: (optional) NXmanipulator

Manipulator for positioning of the sample.

sample_temperature: (recommended) NX_FLOAT

drain_current: (recommended) NX_FLOAT

sample_bias: (recommended) NX_FLOAT

PROCESS: (required) NXprocess

Document an event of data processing, reconstruction, or analysis for this data. Describe the appropriate axis calibrations for your experiment using one or more of the following NXcalibrations
energy_calibration: (optional) NXcalibration
  applied: (required) NX_BOOLEAN
    Has an energy calibration been applied?
  calibrated_axis: (recommended) NX_FLOAT
    This is the calibrated energy axis to be used for data plotting.

angular_calibration: (optional) NXcalibration
  applied: (required) NX_BOOLEAN
    Has an angular calibration been applied?
  calibrated_axis: (recommended) NX_FLOAT
    This is the calibrated angular axis to be used for data plotting.

spatial_calibration: (optional) NXcalibration
  applied: (required) NX_BOOLEAN
    Has an spatial calibration been applied?
  calibrated_axis: (recommended) NX_FLOAT
    This is the calibrated spatial axis to be used for data plotting.

momentum_calibration: (optional) NXcalibration
  applied: (required) NX_BOOLEAN
    Has an momentum calibration been applied?
  calibrated_axis: (recommended) NX_FLOAT
    This is the momentum axis to be used for data plotting.

SAMPLE: (required) NXsample
  name: (required) NX_CHAR
  chemical_formula: (recommended) NX_CHAR
    The chemical formula of the sample. For mixtures use the NXsample_component group in NXsample instead.
  preparation_date: (recommended) NX_DATE_TIME
    Date of preparation of the sample for the XPS experiment (i.e. cleaving, last annealing).
  temperature: (required) NX_FLOAT {units=NX_TEMPERATURE}
    In the case of a fixed temperature measurement this is the scalar temperature of the sample. In the case of an experiment in which the temperature is changed and recoded, this is an array of length m of temperatures. This should be a link to /entry/instrument/manipulator/sample_temperature.
  situation: (required) NX_CHAR
    Any of these values:
    • vacuum
    • inert atmosphere
    • oxidising atmosphere
• reducing atmosphere

gas_pressure: (required) NX_FLOAT

sample_history: (recommended) NXnote

A descriptor to keep track of the treatment of the sample before entering the photoemission experiment. Ideally, a full report of the previous operations, in any format (NXnote allows to add pictures, audio, movies). Alternatively, a reference to the location or a unique identifier or other metadata file. In the case these are not available, free-text description.

preparation_description: (required) NXnote

Description of the surface preparation technique for the XPS experiment, i.e. UHV cleaving, in-situ growth, sputtering/annealing etc. Ideally, a full report of the previous operations, in any format (NXnote allows to add pictures, audio, movies). Alternatively, a reference to the location or a unique identifier or other metadata file. In the case these are not available, free-text description.

DATA: (required) NXdata

@signal: (required) NX_CHAR

Obligatory value: data

data: (required) NX_NUMBER

Represents a measure of one- or more-dimensional photoemission counts, where the varied axis may be for example energy, momentum, spatial coordinate, pump-probe delay, spin index, temperature, etc. The axes traces should be linked to the actual encoder position in NXinstrument or calibrated axes in NXprocess.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

• /NXmpes/ENTRY-group
• /NXmpes/ENTRY/DATA-group
• /NXmpes/ENTRY/DATA/data-field
• /NXmpes/ENTRY/DATA@signal-attribute
• /NXmpes/ENTRY/definition-field
• /NXmpes/ENTRY/definition@version-attribute
• /NXmpes/ENTRY/INSTRUMENT-group
• /NXmpes/ENTRY/INSTRUMENT/BEAM-group
• /NXmpes/ENTRY/INSTRUMENT/BEAM/distance-field
• /NXmpes/ENTRY/INSTRUMENT/BEAM/incident_energy-field
• /NXmpes/ENTRY/INSTRUMENT/BEAM/incident_energy_spread-field
• /NXmpes/ENTRY/INSTRUMENT/BEAM/incident_polarization-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER-group
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/COLLECTIONCOLUMN-group
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/COLLECTIONCOLUMN/contrast_aperture-group
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/COLLECTIONCOLUMN/field_aperture-group
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/COLLECTIONCOLUMN/mode-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/COLLECTIONCOLUMN/projection-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/COLLECTIONCOLUMN/scheme-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/description-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/DETECTOR-group
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/DETECTOR/amplifier_type-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/DETECTOR/DATA-group
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/DETECTOR/DATA/raw-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/DETECTOR/DATA@signal-attribute
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/DETECTOR/detector_type-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/energy_resolution-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/ENERGYDISPERSION-group
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/ENERGYDISPERSION/energy_scan_mode-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/ENERGYDISPERSION/entrance_slit-group
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/ENERGYDISPERSION/exit_slit-group
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/ENERGYDISPERSION/pass_energy-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/ENERGYDISPERSION/scheme-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/fast_axes-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/MANIPULATOR-group
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/MANIPULATOR/drain_current-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/MANIPULATOR/sample_bias-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/MANIPULATOR/sample_temperature-field
• /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYSER/slow_axes-field
• /NXmpes/ENTRY/INSTRUMENT/energy_resolution-field
• /NXmpes/ENTRY/INSTRUMENT/SOURCE-group
• /NXmpes/ENTRY/INSTRUMENT/SOURCE/name-field
• /NXmpes/ENTRY/INSTRUMENT/SOURCE/probe-field
• /NXmpes/ENTRY/INSTRUMENT/SOURCE/type-field
• /NXmpes/ENTRY/PROCESS-group
• /NXmpes/ENTRY/PROCESS/angular_calibration-group
• /NXmpes/ENTRY/PROCESS/angular_calibration/applied-field
• /NXmpes/ENTRY/PROCESS/angular_calibration/calibrated_axis-field
• /NXmpes/ENTRY/PROCESS/energy_calibration-group
• /NXmpes/ENTRY/PROCESS/energy_calibration/applied-field
 NXmpes/ENTRY/PROCESS/energy_calibration/calibrated_axis-field
• /NXmpes/ENTRY/PROCESS/momentum_calibration-group
• /NXmpes/ENTRY/PROCESS/momentum_calibration/applied-field
• /NXmpes/ENTRY/PROCESS/momentum_calibration/calibrated_axis-field
• /NXmpes/ENTRY/PROCESS/spatial_calibration-group
• /NXmpes/ENTRY/PROCESS/spatial_calibration/applied-field
• /NXmpes/ENTRY/PROCESS/spatial_calibration/calibrated_axis-field
• /NXmpes/ENTRY/SAMPLE-group
• /NXmpes/ENTRY/SAMPLE/chemical_formula-field
• /NXmpes/ENTRY/SAMPLE/gas_pressure-field
• /NXmpes/ENTRY/SAMPLE/name-field
• /NXmpes/ENTRY/SAMPLE/preparation_date-field
• /NXmpes/ENTRY/SAMPLE/preparation_description-group
• /NXmpes/ENTRY/SAMPLE/sample_history-group
• /NXmpes/ENTRY/SAMPLE/situation-field
• /NXmpes/ENTRY/SAMPLE/temperature-field
• /NXmpes/ENTRY/start_time-field
• /NXmpes/ENTRY/title-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmpes.nxdl.xml

NXoptical_system_em

Status:
base class, extends NXobject

Description:
A container for qualifying an electron optical system.

Symbols:
No symbol table

Groups cited:
none

Structure:

camera_length: (optional) NX_NUMBER {units=NX_LENGTH}
magnification: (optional) NX_NUMBER {units=NX_DIMENSIONLESS}
defocus: (optional) NX_NUMBER {units=NX_LENGTH}
semi_convergence_angle: (optional) NX_NUMBER {units=NX_ANGLE}
field_of_view: (optional) NX_NUMBER {units=NX_LENGTH}
working_distance: (optional) NX_FLOAT {units=NX_LENGTH}
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXoptical_system_em/camera_length-field
- /NXoptical_system_em/defocus-field
- /NXoptical_system_em/field_of_view-field
- /NXoptical_system_em/magnification-field
- /NXoptical_system_em/semi_convergence_angle-field
- /NXoptical_system_em/working_distance-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXoptical_system_em.nxdl.xml

NXpeak

Status:
base class, extends NXobject

Description:
Description of peaks, their functional form or measured support.

Symbols:
The symbols used in the schema to specify e.g. dimensions of arrays.

- n_support: Number of support points

Groups cited:
NXcollection

Structure:

- label: (optional) NX_CHAR
  Human-readable identifier to specify which concept/entity the peak represents/identifies.

- peak_model: (optional) NX_CHAR
  Is the peak described analytically via a functional form or is it empirically defined via measured/reported intensity/counts as a function of an independent variable.
  
  If the functional form is not empirical or gaussian, users should enter other for the peak_model and add relevant details in the NXcollection.
  
  Any of these values: empirical | gaussian | lorentzian | other

- position: (optional) NX_NUMBER (Rank: 1, Dimensions: [n_support]) {units=NX_ANY}
  In the case of an empirical description of the peak and its shoulders, this array holds the position values for the independent variable.

- intensity: (optional) NX_NUMBER (Rank: 1, Dimensions: [n_support]) {units=NX_ANY}
  In the case of an empirical description of the peak and its shoulders, this array holds the intensity/count values at each position.

- COLLECTION: (optional) NXcollection
In the case of an analytical description (or if peak_model is other) this collection holds parameter of (and eventually) the functional form. For example in the case of Gaussians mu, sigma, cut-off values, and background intensity are relevant parameter.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXpeak/COLLECTION-group`
- `/NXpeak/intensity-field`
- `/NXpeak/label-field`
- `/NXpeak/peak_model-field`
- `/NXpeak/position-field`

**NXDL Source:**
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXpeak.nxdl.xml

**NXpulser_apm**

**Status:**
base class, extends `NXobject`

**Description:**
Metadata for laser-, voltage-, or combined pulsing triggering field evaporation.

**Symbols:**
The symbols used in the schema to specify e.g. dimensions of arrays.

- `n_ions`: Total number of ions collected.

**Groups cited:**
`NXbeam, NXcollection, NXmanufacturer, NXsource, NXtransformations`

**Structure:**

- `pulse_mode`: (optional) `NX_CHAR`
  How is field evaporation physically triggered.
  Any of these values: `laser | high_voltage | laser_and_high_voltage`

- `pulse_frequency`: (optional) `NX_NUMBER` `{units=NX_FREQUENCY}`
  Frequency with which the high voltage or laser pulser fires.

- `pulse_fraction`: (optional) `NX_NUMBER` `{units=NX_UNITLESS}`
  Fraction of the pulse_voltage that is applied in addition to the standing_voltage at peak voltage of a pulse.

- `pulsed_voltage`: (optional) `NX_FLOAT` (Rank: 1, Dimensions: [n_ions]) `{units=NX_VOLTAGE}`

- `standing_voltage`: (optional) `NX_FLOAT` (Rank: 1, Dimensions: [n_ions]) `{units=NX_VOLTAGE}`
  Direct current voltage between the specimen and the (local electrode) in the case of local electrode atom probe (LEAP) instrument.

- `laser-gun`: (optional) `NXsource`
Atom probe microscopes use controlled laser, voltage, or a combination of pulsing strategies to trigger the excitation and eventual field evaporation/emission of an ion during an atom probe microscopy experiment.

**name**: (optional) **NX_CHAR**

Given name/alias.

**wavelength**: (optional) **NX_FLOAT** {units=NX_WAVELENGTH}

Nominal wavelength of the laser radiation.

**power**: (optional) **NX_FLOAT** {units=NX_POWER}

Average power of the laser source while illuminating the specimen.

**pulse_energy**: (optional) **NX_FLOAT** {units=NX_ENERGY}

Average energy of the laser at peak of each pulse.

**MANUFACTURER**: (optional) **NXmanufacturer**

**TRANSFORMATIONS**: (optional) **NXtransformations**

Affine transformations which describe the geometry how the laser focusing optics/pinhole-attached coordinate system is defined, how it has to be transformed so that it aligns with the specimen coordinate system. A right-handed Cartesian coordinate system, the so-called laser space, should be assumed, whose positive z-axis points into the direction of the propagating laser beam.

**laser_beam**: (optional) **NXbeam**

Details about specific positions along the focused laser beam which illuminates the (atom probe) specimen.

**pinhole_position**: (optional) **NXcollection**

Track time-dependent settings over the course of the measurement where the laser beam exits the focusing optics.

**spot_position**: (optional) **NXcollection**

Track time-dependent settings over the course of the measurement where the laser hits the specimen.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXpulser_apm/laser_beam-group
- /NXpulser_apm/laser_beam/pinhole_position-group
- /NXpulser_apm/laser_beam/spot_position-group
- /NXpulser_apm/laser_gun-group
- /NXpulser_apm/laser_gun/MANUFACTURER-group
- /NXpulser_apm/laser_gun/name-field
- /NXpulser_apm/laser_gun/power-field
- /NXpulser_apm/laser_gun/pulse_energy-field
- /NXpulser_apm/laser_gun/TRANSFORMATIONS-group
• /NXpulser_apm/laser_gun/wavelength-field
• /NXpulser_apm/pulse_fraction-field
• /NXpulser_apm/pulse_frequency-field
• /NXpulser_apm/pulse_mode-field
• /NXpulser_apm/pulsed_voltage-field
• /NXpulser_apm/standing_voltage-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXpulser_apm.nxdl.xml

NXpump

Status:
base class, extends NXobject

Description:
Device to reduce an atmosphere to a controlled remaining pressure level.

Symbols:
No symbol table

Groups cited:
one

Structure:

**design**: (optional) **NX_CHAR**
Principle type of the pump.
Any of these values:
• membrane
• rotary_vane
• roots
• turbo_molecular

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.
• /NXpump/design-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXpump.nxdl.xml
NXquadric

Status:
   base class, extends NXobject

Description:
   definition of a quadric surface.

Symbols:
   No symbol table

Groups cited:
   none

Structure:
   parameters: (optional) NX_NUMBER (Rank: 1, Dimensions: [10]) {units=NX_PER_LENGTH}
      Ten real values of the matrix that defines the quadric surface in projective space. Ordered Q11, Q12, Q13, Q22, Q23, Q33, P1, P2, P3, R. Takes a units attribute of dimension reciprocal length. R is scalar. P has dimension reciprocal length, and the given units. Q has dimension reciprocal length squared, and units the square of those given.

   surface_type: (optional) NX_CHAR
      An optional description of the form of the quadric surface:
      Any of these values:
      • ELLIPSOID
      • ELLIPTIC_PARABOLOID
      • HYPERBOLIC_PARABOLOID
      • ELLIPTIC_HYPERBOLOID_OF_1_SHEET
      • ELLIPTIC_HYPERBOLOID_OF_2_SHEETS
      • ELLIPTIC_CONE
      • ELLIPTIC_CYLINDER
      • HYPERBOLIC_CYLINDER
      • PARABOLIC_CYLINDER
      • SPHEROID
      • SPHERE
      • PARABOloid
      • HYPERBOLOID_1_SHEET
      • HYPERBOLOID_2_SHEET
      • CONE
      • CYLINDER
      • PLANE
      • IMAGINARY
      • UNKNOWN
depends_on: (optional) \textit{NX_CHAR}

Path to an \textit{NXtransformations} that defining the axis on which the orientation of the surface depends.

\textbf{Hypertext Anchors}

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXquadric/depends_on-field
- /NXquadric/parameters-field
- /NXquadric/surface_type-field

\textbf{NXDL Source:}

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXquadric.nxdl.xml

\textit{NXquadrupole_magnet}

\textbf{Status:}

base class, extends \textit{NXobject}

\textbf{Description:}

definition for a quadrupole magnet.

\textbf{Symbols:}

No symbol table

\textbf{Groups cited:}

\textit{NXlog}

\textbf{Structure:}

- \textit{description}: (optional) \textit{NX_CHAR}
  
  extended description of the magnet.

- \textit{beamline_distance}: (optional) \textit{NX_FLOAT} \{units=\textit{NX_LENGTH}\}
  
  define position of beamline element relative to production target

- \textit{set_current}: (optional) \textit{NX_FLOAT} \{units=\textit{NX_CURRENT}\}
  
  current set on supply.

- \textit{read_current}: (optional) \textit{NXlog}
  
  current read from supply.

  \textit{value}: (optional) \textit{NX_CHAR} \{units=\textit{NX_CURRENT}\}

- \textit{read_voltage}: (optional) \textit{NXlog}
  
  voltage read from supply.

  \textit{value}: (optional) \textit{NX_CHAR} \{units=\textit{NX_VOLTAGE}\}
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXquadrupole_magnet/beamline_distance-field
- /NXquadrupole_magnet/description-field
- /NXquadrupole_magnet/read_current-group
- /NXquadrupole_magnet/read_current/value-field
- /NXquadrupole_magnet/read_voltage-group
- /NXquadrupole_magnet/read_voltage/value-field
- /NXquadrupole_magnet/set_current-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXquadrupole_magnet.nxdl.xml

NXreflectron

Status:
base class, extends NXobject

Description:
Device for reducing flight time differences of ions in ToF experiments.

Symbols:
No symbol table

Groups cited:
NXmanufacturer, NXtransformations

Structure:

name: (optional) NX_CHAR
Given name/alias.

description: (optional) NX_CHAR
Free-text field to specify further details about the reflectron. The field can be used to inform e.g. if the reflectron is flat or curved.

MANUFACTURER: (optional) NXmanufacturer

TRANSFORMATIONS: (optional) NXtransformations
Affine transformation(s) which detail where the reflectron is located relative to e.g. the origin of the specimen space reference coordinate system. This group can also be used for specifying how the reflectron is rotated relative to the specimen axis. The purpose of these more detailed instrument descriptions is to support the creation of a digital twin of the instrument for computational science.
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- `/NXreflectron/description-field`
- `/NXreflectron/MANUFACTURER-group`
- `/NXreflectron/name-field`
- `/NXreflectron/TRANSFORMATIONS-group`

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXreflectron.nxdl.xml

NXregion

Status:

base class, extends NXobject

Description:

Geometry and logical description of a region of data in a parent group. When used, it could be a child group to, say, NXdetector.

This can be used to describe a subset of data used to create downsampled data or to derive some data from that subset.

Note, the fields for the rectangular region specifiers follow HDF5’s dataspace hyperslab parameters (see https://portal.hdfgroup.org/display/HDF5/H5S_SELECT_HYPERSLAB). Note when block = 1, then stride ≡ step in Python slicing.

For example, a ROI sum of an area starting at index of [20,50] and shape [220,120] in image data:

```python
detector:NXdetector/
  data[60,256,512]
region:NXregion/
  @region_type = "rectangular"
  parent = "data"
  start = [20,50]
  count = [220,120]
statistics:NXdata/
  @signal = "sum"
  sum[60]
```

the sum dataset contains the summed areas in each frame. Another example, a hyperspectral image downsampled 16-fold in energy:

```python
detector:NXdetector/
  data[128,128,4096]
region:NXregion/
  @region_type = "rectangular"
  parent = "data"
  start = [2]
  count = [20]
  stride = [32]
  block = [16]
```

(continues on next page)
the copy dataset selects 20 16-channel blocks that start 32 channels apart, the maximum dataset will show maximum values in each 16-channel block in every spectra.

Symbols:

These symbols will denote how the shape of the parent group’s data field,

\[(D_0, ..., D_{O-1}, d_0, ..., d_{R-1})\]

could be split into a left set of \(O\) outer dimensions, \(D\), and a right set of \(R\) region dimensions, \(d\), where the data field has rank \(O + R\). Note \(O >= 0\).

\(O\): Outer rank

\(R\): Region rank

Groups cited:

*NXdata*

Structure:

@region_type: (required) *NX_CHAR*

This is rectangular to describe the region as a hyper-rectangle in the index space of its parent group’s data field.

Obligatory value: rectangular

parent: (optional) *NX_CHAR*

The name of data field in the parent group or the path of a data field relative to the parent group (so it could be a field in a subgroup of the parent group)

parent_mask: (optional) *NX_CHAR*

The name of an optional mask field in the parent group with rank \(R\) and dimensions \(d\). For example, this could be pixel_mask of an NXdetector.

start: (recommended) *NX_NUMBER* (Rank: 1, Dimensions: \([R]\))

The starting position for region in detector data field array. This is recommended as it also defines the region rank. If omitted then defined as an array of zeros.

count: (recommended) *NX_INT* (Rank: 1, Dimensions: \([R]\))

The number of blocks or items in the hyperslab selection. If omitted then defined as an array of dimensions that take into account the other hyperslab selection fields to span the parent data field’s shape.

stride: (optional) *NX_INT* (Rank: 1, Dimensions: \([R]\))

An optional field to define striding used to downsample data. If omitted then defined as an array of ones.

block: (optional) *NX_INT* (Rank: 1, Dimensions: \([R]\))
An optional field to define the block size used to copy or downsample data. In the $i$-th dimension, if $\text{block}[i] < \text{stride}[i]$ then the downsampling blocks have gaps between them; when $\text{block}$ matches $\text{stride}$ then the blocks are contiguous; otherwise the blocks overlap. If omitted then defined as an array of ones.

**scale:** (optional) *NX_NUMBER* (Rank: 1, Dimensions: [R])

An optional field to define a divisor for scaling of reduced data. For example, in a downsampled sum, it can reduce the maximum values to fit in the domain of the result data type. In an image that is downsampled by summing 2x2 blocks, using $\text{scale} = 4$ allows the result to fit in the same integer type dataset as the parent dataset. If omitted then no scaling occurs.

**downsampled:** (optional) *NXdata*

An optional group containing data copied/downsampled from parent group’s data. Its dataset name must reflect how the downsampling is done over each block. So it could be a reduction operation such as sum, minimum, maximum, mean, mode, median, etc. If downsampling is merely copying each block then use “copy” as the name. Where more than one downsample dataset is written (specified with @signal) then add @auxiliary_signals listing the others. In the copy case, the field should have a shape of $(D_0, ..., D_{O-1}, \text{block}[0] \times \text{count}[0], ..., \text{block}[R-1] \times \text{count}[R-1])$, otherwise the expected shape is $(D_0, ..., D_{O-1}, \text{count}[0], ..., \text{count}[R-1])$.

The following figure shows how blocks are used in downsampling:

![Diagram of downsampling process](image)

Fig. 14: A selection with $\text{start} = 2$, $\text{count} = 4$, $\text{stride} = 3$, $\text{block} = 2$ from a dataset with shape [13] will result in the reduce dataset of shape [4] and a copy dataset of shape [8].

**statistics:** (optional) *NXdata*

An optional group containing any statistics derived from the region in parent group’s data such as sum, minimum, maximum, mean, mode, median, rms, variance, etc. Where more than one statistical dataset is written (specified with @signal) then add @auxiliary_signals listing the others. All data fields should have shapes of $D$. 

### 3.3. NeXus Class Definitions
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXregion/block-field
- /NXregion/count-field
- /NXregion/downsampled-group
- /NXregion/parent-field
- /NXregion/parent_mask-field
- /NXregion/scale-field
- /NXregion/start-field
- /NXregion/statistics-group
- /NXregion/stride-field
- /NXregion@region_type-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXregion.nxdl.xml

NXregistration

Status:
base class, extends NXobject

Description:
Describes image registration procedures.

Symbols:
No symbol table

Groups cited:
NXtransformations

Structure:
- applied: (optional) NX_BOOLEAN
  Has the registration been applied?
- last_process: (optional) NX_CHAR
  Indicates the name of the last operation applied in the NXprocess sequence.
- depends_on: (optional) NX_CHAR
  Specifies the position by pointing to the last transformation in the transformation chain in the
  NXtransformations group.
- description: (optional) NX_CHAR
  Description of the procedures employed.

TRANSFORMATIONS: (optional) NXtransformations
To describe the operations of image registration (combinations of rigid translations and rotations)
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXregistration/applied-field
- /NXregistration/depends_on-field
- /NXregistration/description-field
- /NXregistration/last_process-field
- /NXregistration/TRANSFORMATIONS-group

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXregistration.nxdl.xml

NXscanbox_em

Status:
base class, extends NXobject

Description:
Scan box and coils which deflect an electron beam in a controlled manner.
In electron microscopy, the scan box is instructed by the microscope control software. This component
directs the probe to controlled locations according to a scan scheme and plan.

Symbols:
No symbol table

Groups cited:
none

Structure:

- calibration_style: (optional) NX_CHAR
- center: (optional) NX_NUMBER {units=NX_ANY}
- flyback_time: (optional) NX_FLOAT {units=NX_TIME}
- line_time: (optional) NX_FLOAT {units=NX_TIME}
- pixel_time: (optional) NX_FLOAT {units=NX_TIME}
- requested_pixel_time: (optional) NX_FLOAT {units=NX_TIME}
- rotation: (optional) NX_FLOAT {units=NX_ANGLE}
- ac_line_sync: (optional) NX_BOOLEAN
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXscanbox_em/ac_line_sync-field
- /NXscanbox_em/calibration_style-field
- /NXscanbox_em/center-field
- /NXscanbox_em/flyback_time-field
- /NXscanbox_em/line_time-field
- /NXscanbox_em/pixel_time-field
- /NXscanbox_em/requested_pixel_time-field
- /NXscanbox_em/rotation-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXscanbox_em.nxdl.xml

NXseparator

Status:
- base class, extendsNXobject

Description:
definition for an electrostatic separator.

Symbols:
- No symbol table

Groups cited:
- NXlog

Structure:

description: (optional) NX_CHAR
extended description of the separator.

beamline_distance: (optional) NX_FLOAT {units=NX_LENGTH}
define position of beamline element relative to production target

set_Bfield_current: (optional) NX_FLOAT {units=NX_CURRENT}
current set on magnet supply.

set_Efield_voltage: (optional) NX_FLOAT {units=NX_VOLTAGE}
current set on HT supply.

read_Bfield_current: (optional) NXlog
current read from magnet supply.

value: (optional) NX_CHAR {units=NX_CURRENT}

read_Bfield_voltage: (optional) NXlog
voltage read from magnet supply.

value: (optional) \textit{NX\_CHAR \{units=NX\_VOLTAGE\}}

\textbf{read\_Efield\_current}: (optional) \textit{NXlog}

current read from HT supply.

value: (optional) \textit{NX\_CHAR \{units=NX\_CURRENT\}}

\textbf{read\_Efield\_voltage}: (optional) \textit{NXlog}

voltage read from HT supply.

value: (optional) \textit{NX\_CHAR \{units=NX\_VOLTAGE\}}

\section*{Hypertext Anchors}

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- \textit{/NXseparator/beamline\_distance-field}
- \textit{/NXseparator/description-field}
- \textit{/NXseparator/read\_Bfield\_current-group}
- \textit{/NXseparator/read\_Bfield\_current/value-field}
- \textit{/NXseparator/read\_Bfield\_voltage-group}
- \textit{/NXseparator/read\_Bfield\_voltage/value-field}
- \textit{/NXseparator/read\_Efield\_current-group}
- \textit{/NXseparator/read\_Efield\_current/value-field}
- \textit{/NXseparator/read\_Efield\_voltage-group}
- \textit{/NXseparator/read\_Efield\_voltage/value-field}
- \textit{/NXseparator/set\_Bfield\_current-field}
- \textit{/NXseparator/set\_Efield\_voltage-field}

\textbf{NXDL Source:}

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXseparator.nxdl.xml

\textbf{NXsnnsevent}

\textbf{Status:}

application definition, extends \textit{NXobject}

\textbf{Description:}

This is a definition for event data from Spallation Neutron Source (SNS) at ORNL.

\textbf{Symbols:}

No symbol table

\textbf{Groups cited:}

\textit{NXaperture, NXattenuator, NXcollection, NXcrystal, NXdata, NXdetector, NXdisk\_chopper, NXentry, NXevent\_data, NXgeometry, NXinstrument, NXlog, NXmoderator, NXmonitor, NXnote, NXorientation, NXPolarizer, NXpositioner, NXsample, NXshape, NXsource, NXtranslation, NXuser}
Structure:

**ENTRY**: (required) NXentry

- **collection_identifier**: (required) NX_CHAR
- **collection_title**: (required) NX_CHAR
- **definition**: (required) NX_CHAR
  
  Official NXDL schema after this file goes to applications.

  Obligatory value: NXsnsevent

- **duration**: (required) NX_FLOAT {units=NX_TIME}
- **end_time**: (required) NX_DATE_TIME
- **entry_identifier**: (required) NX_CHAR
- **experiment_identifier**: (required) NX_CHAR
- **notes**: (required) NX_CHAR
- **proton_charge**: (required) NX_FLOAT {units=NX_CHARGE}
- **raw_frames**: (required) NX_INT
- **run_number**: (required) NX_CHAR
- **start_time**: (required) NX_DATE_TIME
- **title**: (required) NX_CHAR
- **total_counts**: (required) NX_UINT {units=NX_UNITLESS}
- **total_uncounted_counts**: (required) NX_UINT {units=NX_UNITLESS}

**DASlogs**: (required) NXcollection

  Details of all logs, both from cvinfo file and from HistoTool (frequency and proton_charge).

- **LOG**: (required) NXlog

  - **average_value**: (required) NX_FLOAT
  - **average_value_error**: (optional) NX_FLOAT

    DEPRECATED: see https://github.com/nexusformat/definitions/issues/821

  - **average_value_errors**: (required) NX_FLOAT
  - **description**: (required) NX_CHAR
  - **duration**: (required) NX_FLOAT
  - **maximum_value**: (required) NX_FLOAT
  - **minimum_value**: (required) NX_FLOAT
  - **time**: (required) NX_FLOAT (Rank: 1, Dimensions: [nvalue])
  - **value**: (required) NX_FLOAT (Rank: 1, Dimensions: [nvalue])

**POSITIONER**: (optional) NXpositioner
Motor logs from cvinfo file.

average_value: (required) NX_FLOAT

average_value_error: (optional) NX_FLOAT

DEPRECATED: see https://github.com/nexusformat/definitions/issues/821

average_value_errors: (required) NX_FLOAT
description: (required) NX_CHAR
duration: (required) NX_FLOAT
maximum_value: (required) NX_FLOAT
minimum_value: (required) NX_FLOAT
time: (required) NX_FLOAT (Rank: 1, Dimensions: [numvalue])
value: (required) NX_FLOAT (Rank: 1, Dimensions: [numvalue])

SNSHistToool: (required) NXnote

SNSbanking_file_name: (required) NX_CHAR
SNSmapping_file_name: (required) NX_CHAR
author: (required) NX_CHAR
command1: (required) NX_CHAR

Command string for event2nxl.
date: (required) NX_CHAR
description: (required) NX_CHAR
version: (required) NX_CHAR

DATA: (required) NXdata
data_x_y: link (suggested target: /NXentry/NXinstrument/NXdetector/data_x_y)
x_pixel_offset: link (suggested target: /NXentry/NXinstrument/NXdetector/x_pixel_offset)
y_pixel_offset: link (suggested target: /NXentry/NXinstrument/NXdetector/y_pixel_offset)

EVENT_DATA: (required) NXevent_data
event_index: link (suggested target: /NXentry/NXinstrument/NXdetector/event_index)
event_pixel_id: link (suggested target: /NXentry/NXinstrument/NXdetector/event_pixel_id)
event_time_of_flight: link (suggested target: /NXentry/NXinstrument/NXdetector/event_time_of_flight)
pulse_time: link (suggested target: /NXentry/NXinstrument/NXdetector/pulse_time)
instrument: (required) NXinstrument

SNSdetector_calibration_id: (required) NX_CHAR
Detector calibration id from DAS.

**SNSgeometry_file_name**: (required) *NX_CHAR*

**SNStranslation_service**: (required) *NX_CHAR*

**beamline**: (required) *NX_CHAR*

**name**: (required) *NX_CHAR*

**SNS**: (required) *NXsource*

  **frequency**: (required) *NX_FLOAT* {units=*NX_FREQUENCY*}

  **name**: (required) *NX_CHAR*

  **probe**: (required) *NX_CHAR*

  **type**: (required) *NX_CHAR*

**DETECTOR**: (required) *NXdetector*

  **azimuthal_angle**: (required) *NX_FLOAT* (Rank: 2, Dimensions: [numx, numy]) {units=*NX_ANGLE*}

  **data_x_y**: (required) *NX_UINT* (Rank: 2, Dimensions: [numx, numy])

    *expect signal=2 axes="x_pixel_offset,y_pixel_offset"

  **distance**: (required) *NX_FLOAT* (Rank: 2, Dimensions: [numx, numy]) {units=*NX_LENGTH*}

  **event_index**: (required) *NX_UINT* (Rank: 1, Dimensions: [numpulses])

  **event_pixel_id**: (required) *NX_UINT* (Rank: 1, Dimensions: [numevents])

  **event_time_of_flight**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [numevents]) {units=*NX_TIME_OF_FLIGHT*}

  **pixel_id**: (required) *NX_UINT* (Rank: 2, Dimensions: [numx, numy])

  **polar_angle**: (required) *NX_FLOAT* (Rank: 2, Dimensions: [numx, numy]) {units=*NX_ANGLE*}

  **pulse_time**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [numpulses]) {units=*NX_TIME*}

  **total_counts**: (required) *NX_UINT*

  **x_pixel_offset**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [numx]) {units=*NX_LENGTH*}

  **y_pixel_offset**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [numy]) {units=*NX_LENGTH*}

  **origin**: (required) *NXgeometry*

**orientation**: (required) *NXorientation*

  **value**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [6])

    *Six out of nine rotation parameters.*

**shape**: (required) *NXshape*

  **description**: (required) *NX_CHAR*

  **shape**: (required) *NX_CHAR*
size: (required) $NX_FLOAT$ (Rank: 1, Dimensions: [3]) [units=$NX_LENGTH$]

translation: (required) $NX_translation$

distance: (required) $NX_FLOAT$ (Rank: 1, Dimensions: [3]) [units=$NX_LENGTH$]

DISK_CHOPPER: (optional) $NX_disk_chopper$

distance: (required) $NX_FLOAT$ [units=$NX_LENGTH$]

moderator: (required) $NX_moderator$

coupling_material: (required) $NX_CHAR$

distance: (required) $NX_FLOAT$ [units=$NX_LENGTH$]

temperature: (required) $NX_FLOAT$ [units=$NX_TEMPERATURE$]

type: (required) $NX_CHAR$

APERTURE: (optional) $NX_aperture$

x_pixel_offset: (required) $NX_FLOAT$ [units=$NX_LENGTH$]

origin: (required) $NX_geometry$

orientation: (required) $NX_orientation$

value: (required) $NX_FLOAT$ (Rank: 1, Dimensions: [6])

Six out of nine rotation parameters.

shape: (required) $NX_shape$

description: (required) $NX_CHAR$

shape: (required) $NX_CHAR$

size: (required) $NX_FLOAT$ (Rank: 1, Dimensions: [3]) [units=$NX_LENGTH$]

translation: (required) $NX_translation$

distance: (required) $NX_FLOAT$ (Rank: 1, Dimensions: [3]) [units=$NX_LENGTH$]

ATTENUATOR: (optional) $NX_attenuator$

distance: (required) $NX_FLOAT$ [units=$NX_LENGTH$]

POLARIZER: (optional) $NX_polarizer$

CRYSTAL: (optional) $NX_crystal$

type: (required) $NX_CHAR$

wavelength: (required) $NX_FLOAT$ [units=$NX_WAVELENGTH$]

origin: (required) $NX_geometry$

description: (required) $NX_CHAR$

orientation: (required) $NX_orientation$

value: (required) $NX_FLOAT$ (Rank: 1, Dimensions: [6])

Six out of nine rotation parameters.
shape: (required) NXshape

description: (required) NX_CHAR

shape: (required) NX_CHAR

size: (required) NX_FLOAT {units=NX_LENGTH}

translation: (required) NXtranslation

distance: (required) NX_FLOAT (Rank: 1, Dimensions: [3])
{units=NX_LENGTH}

MONITOR: (optional) NXmonitor

data: (required) NX_UINT (Rank: 1, Dimensions: [numtimechannels])

expect signal=1 axes="time_of_flight"

distance: (required) NX_FLOAT {units=NX_LENGTH}

mode: (required) NX_CHAR

time_of_flight: (required) NX_FLOAT (Rank: 1, Dimensions: [numtimechannels + 1])
{units=NX_TIME}

sample: (required) NXsample

changer_position: (required) NX_CHAR

holder: (required) NX_CHAR

identifier: (required) NX_CHAR

name: (required) NX_CHAR

Descriptive name of sample

nature: (required) NX_CHAR

USER: (required) NXuser

facility_user_id: (required) NX_CHAR

name: (required) NX_CHAR

role: (required) NX_CHAR

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXsnsevent/ENTRY-group
- /NXsnsevent/ENTRY/collection_identifier-field
- /NXsnsevent/ENTRY/collection_title-field
- /NXsnsevent/ENTRY/DASlogs-group
- /NXsnsevent/ENTRY/DASlogs/LOG-group
- /NXsnsevent/ENTRY/DASlogs/LOG/average_value-field
- /NXsnsevent/ENTRY/DASlogs/LOG/average_value_error-field
- /NXsnsevent/ENTRY/DASlogs/LOG/average_value_errors-field
- /NXsnsevent/ENTRY/DASlogs/LOG/description-field
- /NXsnsevent/ENTRY/DASlogs/LOG/duration-field
- /NXsnsevent/ENTRY/DASlogs/LOG/maximum_value-field
- /NXsnsevent/ENTRY/DASlogs/LOG/minimum_value-field
- /NXsnsevent/ENTRY/DASlogs/LOG/time-field
- /NXsnsevent/ENTRY/DASlogs/LOG/value-field
- /NXsnsevent/ENTRY/DASlogs/POSITIONER-group
- /NXsnsevent/ENTRY/DASlogs/POSITIONER/average_value-field
- /NXsnsevent/ENTRY/DASlogs/POSITIONER/average_value_error-field
- /NXsnsevent/ENTRY/DASlogs/POSITIONER/average_value_errors-field
- /NXsnsevent/ENTRY/DASlogs/POSITIONER/description-field
- /NXsnsevent/ENTRY/DASlogs/POSITIONER/duration-field
- /NXsnsevent/ENTRY/DASlogs/POSITIONER/maximum_value-field
- /NXsnsevent/ENTRY/DASlogs/POSITIONER/minimum_value-field
- /NXsnsevent/ENTRY/DASlogs/POSITIONER/time-field
- /NXsnsevent/ENTRY/DASlogs/POSITIONER/value-field
- /NXsnsevent/ENTRY/DATA-group
- /NXsnsevent/ENTRY/DATA/data_x_y-link
- /NXsnsevent/ENTRY/DATA/x_pixel_offset-link
- /NXsnsevent/ENTRY/DATA/y_pixel_offset-link
- /NXsnsevent/ENTRY/definition-field
- /NXsnsevent/ENTRY/duration-field
- /NXsnsevent/ENTRY/end_time-field
- /NXsnsevent/ENTRY/entry_identifier-field
- /NXsnsevent/ENTRY/EVENT_DATA-group
- /NXsnsevent/ENTRY/EVENT_DATA/event_index-link
- /NXsnsevent/ENTRY/EVENT_DATA/event_pixel_id-link
- /NXsnsevent/ENTRY/EVENT_DATA/event_time_of_flight-link
- /NXsnsevent/ENTRY/EVENT_DATA/pulse_time-link
- /NXsnsevent/ENTRY/experiment_identifier-field
- /NXsnsevent/ENTRY/instrument-group
- /NXsnsevent/ENTRY/instrument/APERTURE-group
- /NXsnsevent/ENTRY/instrument/APERTURE/origin-group
- /NXsnsevent/ENTRY/instrument/APERTURE/origin/orientation-group
- /NXsnsevent/ENTRY/instrument/APERTURE/origin/orientation/value-field
- /NXsnsevent/ENTRY/instrument/APERTURE/origin/shape-group
- /NXsevent/ENTRY/instrument/APERTURE/origin/shape/description-field
- /NXsevent/ENTRY/instrument/APERTURE/origin/shape/field
- /NXsevent/ENTRY/instrument/APERTURE/origin/size-field
- /NXsevent/ENTRY/instrument/APERTURE/origin/translation/group
- /NXsevent/ENTRY/instrument/APERTURE/origin/translation/distance-field
- /NXsevent/ENTRY/instrument/APERTURE/x_pixel_offset-field
- /NXsevent/ENTRY/instrument/ATTENUATOR/group
- /NXsevent/ENTRY/instrument/ATTENUATOR/distance-field
- /NXsevent/ENTRY/instrument/beamline-field
- /NXsevent/ENTRY/instrument/CRYSTAL/group
- /NXsevent/ENTRY/instrument/CRYSTAL/origin/group
- /NXsevent/ENTRY/instrument/CRYSTAL/origin/description-field
- /NXsevent/ENTRY/instrument/CRYSTAL/origin/orientation/group
- /NXsevent/ENTRY/instrument/CRYSTAL/origin/orientation/value-field
- /NXsevent/ENTRY/instrument/CRYSTAL/origin/shape/group
- /NXsevent/ENTRY/instrument/CRYSTAL/origin/shape/description-field
- /NXsevent/ENTRY/instrument/CRYSTAL/origin/shape/field
- /NXsevent/ENTRY/instrument/CRYSTAL/origin/shape/size-field
- /NXsevent/ENTRY/instrument/CRYSTAL/origin/translation/group
- /NXsevent/ENTRY/instrument/CRYSTAL/origin/translation/distance-field
- /NXsevent/ENTRY/instrument/CRYSTAL/type-field
- /NXsevent/ENTRY/instrument/CRYSTAL/wavelength-field
- /NXsevent/ENTRY/instrument/DETECTOR/group
- /NXsevent/ENTRY/instrument/DETECTOR/azimuthal_angle-field
- /NXsevent/ENTRY/instrument/DETECTOR/data_x_y-field
- /NXsevent/ENTRY/instrument/DETECTOR/distance-field
- /NXsevent/ENTRY/instrument/DETECTOR/event_index-field
- /NXsevent/ENTRY/instrument/DETECTOR/event_pixel_id-field
- /NXsevent/ENTRY/instrument/DETECTOR/event_time_of_flight-field
- /NXsevent/ENTRY/instrument/DETECTOR/origin/group
- /NXsevent/ENTRY/instrument/DETECTOR/origin/orientation/group
- /NXsevent/ENTRY/instrument/DETECTOR/origin/orientation/value-field
- /NXsevent/ENTRY/instrument/DETECTOR/origin/shape/group
- /NXsevent/ENTRY/instrument/DETECTOR/origin/shape/description-field
- /NXsevent/ENTRY/instrument/DETECTOR/origin/shape/field
- /NXsevent/ENTRY/instrument/DETECTOR/origin/shape/size-field
• /NXsnsevent/ENTRY/instrument/DETECTOR/origin/translation-group
• /NXsnsevent/ENTRY/instrument/DETECTOR/origin/translation/distance-field
• /NXsnsevent/ENTRY/instrument/DETECTOR/pixel_id-field
• /NXsnsevent/ENTRY/instrument/DETECTOR/polar_angle-field
• /NXsnsevent/ENTRY/instrument/DETECTOR/pulse_time-field
• /NXsnsevent/ENTRY/instrument/DETECTOR/total_counts-field
• /NXsnsevent/ENTRY/instrument/DETECTOR/x_pixel_offset-field
• /NXsnsevent/ENTRY/instrument/DETECTOR/y_pixel_offset-field
• /NXsnsevent/ENTRY/instrument/DISK_CHOPPER-group
• /NXsnsevent/ENTRY/instrument/DISK_CHOPPER/distance-field
• /NXsnsevent/ENTRY/instrument/moderator-group
• /NXsnsevent/ENTRY/instrument/moderator/coupling_material-field
• /NXsnsevent/ENTRY/instrument/moderator/distance-field
• /NXsnsevent/ENTRY/instrument/moderator/temperature-field
• /NXsnsevent/ENTRY/instrument/moderator/type-field
• /NXsnsevent/ENTRY/instrument/name-field
• /NXsnsevent/ENTRY/instrument/POLARIZER-group
• /NXsnsevent/ENTRY/instrument/SNS-group
• /NXsnsevent/ENTRY/instrument/SNS/frequency-field
• /NXsnsevent/ENTRY/instrument/SNS/name-field
• /NXsnsevent/ENTRY/instrument/SNS/probe-field
• /NXsnsevent/ENTRY/instrument/SNS/type-field
• /NXsnsevent/ENTRY/instrument/SNSdetector_calibration_id-field
• /NXsnsevent/ENTRY/instrument/SNSgeometry_file_name-field
• /NXsnsevent/ENTRY/instrument/SNStranslation_service-field
• /NXsnsevent/ENTRY/MONITOR-group
• /NXsnsevent/ENTRY/MONITOR/data-field
• /NXsnsevent/ENTRY/MONITOR/distance-field
• /NXsnsevent/ENTRY/MONITOR/mode-field
• /NXsnsevent/ENTRY/MONITOR/time_of_flight-field
• /NXsnsevent/ENTRY/notes-field
• /NXsnsevent/ENTRY/proton_charge-field
• /NXsnsevent/ENTRY/raw_frames-field
• /NXsnsevent/ENTRY/run_number-field
• /NXsnsevent/ENTRY/sample-group
• /NXsnsevent/ENTRY/sample/changer_position-field
NXsnsevent

• /NXsnsevent/ENTRY/sample/holder-field
• /NXsnsevent/ENTRY/sample/identifier-field
• /NXsnsevent/ENTRY/sample/name-field
• /NXsnsevent/ENTRY/sample/nature-field
• /NXsnsevent/ENTRY/SNSHistoTool-group
• /NXsnsevent/ENTRY/SNSHistoTool/author-field
• /NXsnsevent/ENTRY/SNSHistoTool/command1-field
• /NXsnsevent/ENTRY/SNSHistoTool/date-field
• /NXsnsevent/ENTRY/SNSHistoTool/description-field
• /NXsnsevent/ENTRY/SNSHistoTool/SNSbanking_file_name-field
• /NXsnsevent/ENTRY/SNSHistoTool/SNSmapping_file_name-field
• /NXsnsevent/ENTRY/SNSHistoTool/version-field
• /NXsnsevent/ENTRY/start_time-field
• /NXsnsevent/ENTRY/title-field
• /NXsnsevent/ENTRY/total_counts-field
• /NXsnsevent/ENTRY/total_uncounted_counts-field
• /NXsnsevent/ENTRY/USER-group
• /NXsnsevent/ENTRY/USER/facility_user_id-field
• /NXsnsevent/ENTRY/USER/name-field
• /NXsnsevent/ENTRY/USER/role-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsnsevent.nxdl.xml

NXsnshisto

Status:
application definition, extends NXobject

Description:
This is a definition for histogram data from Spallation Neutron Source (SNS) at ORNL.

Symbols:
No symbol table

Groups cited:
NXaperture, NXattenuator, NXcollection, NXcrystal,NXdata, NXdetector, NXdisk_chopper, NXentry,
NXfermi_chopper, NXgeometry, NXinstrument, NXlog, NXmoderator, NXmonitor, NXnote, NXorientation, NXpolarizer, NXpositioner, NXsample, NXshape, NXsource, NXtranslation, NXuser

Structure:
ENTRY: (required) NXentry
collection_identifier: (required) `NX_CHAR`
collection_title: (required) `NX_CHAR`
definition: (required) `NX_CHAR`

Official NXDL schema after this file goes to applications.

Obligatory value: `NXsnshisto`
duration: (required) `NX_FLOAT` {units=`NX_TIME`}
end_time: (required) `NX_DATE_TIME`
entry_identifier: (required) `NX_CHAR`
experiment_identifier: (required) `NX_CHAR`
notes: (required) `NX_CHAR`
proton_charge: (required) `NX_FLOAT` {units=`NX_CHARGE`}
raw_frames: (required) `NX_INT`
run_number: (required) `NX_CHAR`
start_time: (required) `NX_DATE_TIME`
title: (required) `NX_CHAR`
total_counts: (required) `NX_UINT` {units=`NX_UNITLESS`}
total_uncounted_counts: (required) `NX_UINT` {units=`NX_UNITLESS`}

DASlogs: (required) `NXcollection`

Details of all logs, both from cvinfo file and from HistoTool (frequency and proton_charge).

LOG: (required) `NXlog`

average_value: (required) `NX_FLOAT`
average_value_error: (optional) `NX_FLOAT`

DEPRECATED: see https://github.com/nexusformat/definitions/issues/821

average_value_errors: (required) `NX_FLOAT`
description: (required) `NX_CHAR`
duration: (required) `NX_FLOAT`
maximum_value: (required) `NX_FLOAT`
minimum_value: (required) `NX_FLOAT`
time: (required) `NX_FLOAT` (Rank: 1, Dimensions: [nvalue])
value: (required) `NX_FLOAT` (Rank: 1, Dimensions: [nvalue])

POSITIONER: (optional) `NXpositioner`

Motor logs from cvinfo file.

average_value: (required) `NX_FLOAT`
average_value_error: (optional) `NX_FLOAT`
**DEPRECATED**: see https://github.com/nexusformat/definitions/issues/821

average_value_errors: (required) `NX_FLOAT`
description: (required) `NX_CHAR`
duration: (required) `NX_FLOAT`
maximum_value: (required) `NX_FLOAT`
minimum_value: (required) `NX_FLOAT`
time: (required) `NX_FLOAT` (Rank: 1, Dimensions: [numvalue])
value: (required) `NX_FLOAT` (Rank: 1, Dimensions: [numvalue])

**SNSHistoTool**: (required) `NXnote`

SNSbanking_file_name: (required) `NX_CHAR`
SNSmapping_file_name: (required) `NX_CHAR`
author: (required) `NX_CHAR`
command1: (required) `NX_CHAR`
  Command string for event2histo_nxl.
date: (required) `NX_CHAR`
description: (required) `NX_CHAR`
version: (required) `NX_CHAR`

**DATA**: (required) `NXdata`

data: `link` (suggested target: /NXentry/NXinstrument/NXdetector/data)
data_x_time_of_flight: `link` (suggested target: /NXentry/NXinstrument/NXdetector/data_x_time_of_flight)
data_x_y: `link` (suggested target: /NXentry/NXinstrument/NXdetector/data_x_y)
data_y_time_of_flight: `link` (suggested target: /NXentry/NXinstrument/NXdetector/data_y_time_of_flight)
pixel_id: `link` (suggested target: /NXentry/NXinstrument/NXdetector/pixel_id)
time_of_flight: `link` (suggested target: /NXentry/NXinstrument/NXdetector/time_of_flight)
total_counts: `link` (suggested target: /NXentry/NXinstrument/NXdetector/total_counts)
x_pixel_offset: `link` (suggested target: /NXentry/NXinstrument/NXdetector/x_pixel_offset)
y_pixel_offset: `link` (suggested target: /NXentry/NXinstrument/NXdetector/y_pixel_offset)

instrument: (required) `NXinstrument`

SNSdetector_calibration_id: (required) `NX_CHAR`
  Detector calibration id from DAS.
**SNSgeometry_file_name**: (required) **NX_CHAR**

**SNSTranslation_service**: (required) **NX_CHAR**

**beamline**: (required) **NX_CHAR**

**name**: (required) **NX_CHAR**

**SNS**: (required) **NXsource**

  **frequency**: (required) **NX_FLOAT** \( \{ \text{units=} \text{NX_FREQUENCY} \} \)

  **name**: (required) **NX_CHAR**

  **probe**: (required) **NX_CHAR**

  **type**: (required) **NX_CHAR**

**DETECTOR**: (required) **NXdetector**

  **azimuthal_angle**: (required) **NX_FLOAT** \( \{ \text{units=} \text{NX_ANGLE} \} \)

  **data**: (required) **NX_UINT** \( \{ \text{Rank=} 2, \text{Dimensions=} \{ \text{numx, numy} \} \} \)

  **data_x_time_of_flight**: (required) **NX_UINT** \( \{ \text{Rank=} 2, \text{Dimensions=} \{ \text{numx, numy, numtof} \} \} \)

  **data_x_y**: (required) **NX_UINT** \( \{ \text{Rank=} 2, \text{Dimensions=} \{ \text{numx, numy} \} \} \)

  **data_y_time_of_flight**: (required) **NX_UINT** \( \{ \text{Rank=} 2, \text{Dimensions=} \{ \text{numy, numtof} \} \} \)

  **distance**: (required) **NX_FLOAT** \( \{ \text{Rank=} 2, \text{Dimensions=} \{ \text{numx, numy} \} \} \)

  **pixel_id**: (required) **NX_UINT** \( \{ \text{Rank=} 2, \text{Dimensions=} \{ \text{numx, numy} \} \} \)

  **polar_angle**: (required) **NX_FLOAT** \( \{ \text{Rank=} 2, \text{Dimensions=} \{ \text{numx, numy} \} \} \)

  **time_of_flight**: (required) **NX_FLOAT** \( \{ \text{Rank=} 1, \text{Dimensions=} \{ \text{numtof + 1} \} \} \)

  **total_counts**: (required) **NX_UINT**

  **x_pixel_offset**: (required) **NX_FLOAT** \( \{ \text{Rank=} 1, \text{Dimensions=} \{ \text{numx} \} \} \)

  **y_pixel_offset**: (required) **NX_FLOAT** \( \{ \text{Rank=} 1, \text{Dimensions=} \{ \text{numy} \} \} \)

**origin**: (required) **NXgeometry**

  **orientation**: (required) **NXorientation**

    **value**: (required) **NX_FLOAT** \( \{ \text{Rank=} 1, \text{Dimensions=} \{ 6 \} \} \)

    Six out of nine rotation parameters.

**shape**: (required) **NXshape**

  **description**: (required) **NX_CHAR**

  **shape**: (required) **NX_CHAR**

  **size**: (required) **NX_FLOAT** \( \{ \text{Rank=} 1, \text{Dimensions=} \{ 3 \} \} \)

\( \{ \text{units=} \text{NX_LENGTH} \} \)
**translation**: (required) *NX_translation*

**distance**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3])
{units=*NX_LENGTH*}

**DISK_CHOPPER**: (optional) *NX_disk_chopper*

Original specification called for NXchopper, which is not a valid NeXus base class. Select either NXdisk_chopper or NXfermi_chopper, as appropriate.

**distance**: (required) *NX_FLOAT* {units=*NX_LENGTH*}

**FERMI_CHOPPER**: (optional) *NX_fermi_chopper*

Original specification called for NXchopper, which is not a valid NeXus base class. Select either NXdisk_chopper or NXfermi_chopper, as appropriate.

**distance**: (required) *NX_FLOAT* {units=*NX_LENGTH*}

**moderator**: (required) *NX_moderator*

**coupling_material**: (required) *NX_CHAR*

**distance**: (required) *NX_FLOAT* {units=*NX_LENGTH*}

**temperature**: (required) *NX_FLOAT* {units=*NX_TEMPERATURE*}

**type**: (required) *NX_CHAR*

**APERTURE**: (optional) *NX_aperture*

**x_pixel_offset**: (required) *NX_FLOAT* {units=*NX_LENGTH*}

**origin**: (required) *NX_geometry*

**orientation**: (required) *NX_orientation*

**value**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [6])

Six out of nine rotation parameters.

**shape**: (required) *NX_shape*

**description**: (required) *NX_CHAR*

**shape**: (required) *NX_CHAR*

**size**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3])
{units=*NX_LENGTH*}

**translation**: (required) *NX_translation*

**distance**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3])
{units=*NX_LENGTH*}

**ATTENUATOR**: (optional) *NX_attenuator*

**distance**: (required) *NX_FLOAT* {units=*NX_LENGTH*}

**POLARIZER**: (optional) *NX_polarizer*

**CRYSTAL**: (optional) *NX_crystal*

**type**: (required) *NX_CHAR*

**wavelength**: (required) *NX_FLOAT* {units=*NX_WAVELENGTH*}

**origin**: (required) *NX_geometry*
description: (required) **NX_CHAR**

orientation: (required) **NX_orientation**

  value: (required) **NX_FLOAT** (Rank: 1, Dimensions: [6])

  Six out of nine rotation parameters.

shape: (required) **NX_shape**

  description: (required) **NX_CHAR**

  shape: (required) **NX_CHAR**

  size: (required) **NX_FLOAT** {units=**NX_LENGTH**}

translation: (required) **NX_translation**

  distance: (required) **NX_FLOAT** (Rank: 1, Dimensions: [3])

  {units=**NX_LENGTH**}

MONITOR: (optional) **NX_monitor**

data: (required) **NX_UINT** (Rank: 1, Dimensions: [numtimechannels])

distance: (required) **NX_FLOAT** {units=**NX_LENGTH**}

mode: (required) **NX_CHAR**

time_of_flight: (required) **NX_FLOAT** (Rank: 1, Dimensions: [numtimechannels + 1]) {units=**NX_TIME**}

sample: (required) **NX_sample**

  changer_position: (required) **NX_CHAR**

  holder: (required) **NX_CHAR**

  identifier: (required) **NX_CHAR**

  name: (required) **NX_CHAR**

    Descriptive name of sample

  nature: (required) **NX_CHAR**

USER: (required) **NX_user**

  facility_user_id: (required) **NX_CHAR**

  name: (required) **NX_CHAR**

  role: (required) **NX_CHAR**

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXsnshisto/ENTRY-group
- /NXsnshisto/ENTRY/collection_identifier-field
- /NXsnshisto/ENTRY/collection_title-field
- /NXsnshisto/ENTRY/DASlogs-group
- /NXsnshisto/ENTRY/DASlogs/LOG-group
• /NXsnshisto/ENTRY/DASlogs/LOG/average_value-field
• /NXsnshisto/ENTRY/DASlogs/LOG/average_value_error-field
• /NXsnshisto/ENTRY/DASlogs/LOG/average_value_errors-field
• /NXsnshisto/ENTRY/DASlogs/LOG/description-field
• /NXsnshisto/ENTRY/DASlogs/LOG/duration-field
• /NXsnshisto/ENTRY/DASlogs/LOG/maximum_value-field
• /NXsnshisto/ENTRY/DASlogs/LOG/minimum_value-field
• /NXsnshisto/ENTRY/DASlogs/LOG/time-field
• /NXsnshisto/ENTRY/DASlogs/LOG/value-field
• /NXsnshisto/ENTRY/DASlogs/POSITIONER-group
• /NXsnshisto/ENTRY/DASlogs/POSITIONER/average_value-field
• /NXsnshisto/ENTRY/DASlogs/POSITIONER/average_value_error-field
• /NXsnshisto/ENTRY/DASlogs/POSITIONER/average_value_errors-field
• /NXsnshisto/ENTRY/DASlogs/POSITIONER/description-field
• /NXsnshisto/ENTRY/DASlogs/POSITIONER/duration-field
• /NXsnshisto/ENTRY/DASlogs/POSITIONER/maximum_value-field
• /NXsnshisto/ENTRY/DASlogs/POSITIONER/minimum_value-field
• /NXsnshisto/ENTRY/DASlogs/POSITIONER/time-field
• /NXsnshisto/ENTRY/DASlogs/POSITIONER/value-field
• /NXsnshisto/ENTRY/DATA-group
• /NXsnshisto/ENTRY/DATA/data-link
• /NXsnshisto/ENTRY/DATA/data_x_time_of_flight-link
• /NXsnshisto/ENTRY/DATA/data_x_y-link
• /NXsnshisto/ENTRY/DATA/data_y_time_of_flight-link
• /NXsnshisto/ENTRY/DATA/pixel_id-link
• /NXsnshisto/ENTRY/DATA/time_of_flight-link
• /NXsnshisto/ENTRY/DATA/total_counts-link
• /NXsnshisto/ENTRY/DATA/x_pixel_offset-link
• /NXsnshisto/ENTRY/DATA/y_pixel_offset-link
• /NXsnshisto/ENTRY/definition-field
• /NXsnshisto/ENTRY/duration-field
• /NXsnshisto/ENTRY/end_time-field
• /NXsnshisto/ENTRY/entry_identifier-field
• /NXsnshisto/ENTRY/experiment_identifier-field
• /NXsnshisto/ENTRY/instrument-group
• /NXsnshisto/ENTRY/instrument/APERTURE-group
• /NXsnshisto/ENTRY/instrument/APERTURE/origin-group
• /NXsnshisto/ENTRY/instrument/APERTURE/origin/orientation-group
• /NXsnshisto/ENTRY/instrument/APERTURE/origin/orientation/value-field
• /NXsnshisto/ENTRY/instrument/APERTURE/origin/shape-group
• /NXsnshisto/ENTRY/instrument/APERTURE/origin/shape/description-field
• /NXsnshisto/ENTRY/instrument/APERTURE/origin/shape/shape-field
• /NXsnshisto/ENTRY/instrument/APERTURE/origin/shape/size-field
• /NXsnshisto/ENTRY/instrument/APERTURE/origin/translation-group
• /NXsnshisto/ENTRY/instrument/APERTURE/origin/translation/distance-field
• /NXsnshisto/ENTRY/instrument/APERTURE/x_pixel_offset-field
• /NXsnshisto/ENTRY/instrument/ATTENUATOR-group
• /NXsnshisto/ENTRY/instrument/ATTENUATOR/distance-field
• /NXsnshisto/ENTRY/instrument/beamline-field
• /NXsnshisto/ENTRY/instrument/CRYSTAL-group
• /NXsnshisto/ENTRY/instrument/CRYSTAL/origin-group
• /NXsnshisto/ENTRY/instrument/CRYSTAL/origin/description-field
• /NXsnshisto/ENTRY/instrument/CRYSTAL/origin/orientation-group
• /NXsnshisto/ENTRY/instrument/CRYSTAL/origin/orientation/value-field
• /NXsnshisto/ENTRY/instrument/CRYSTAL/origin/shape-group
• /NXsnshisto/ENTRY/instrument/CRYSTAL/origin/shape/description-field
• /NXsnshisto/ENTRY/instrument/CRYSTAL/origin/shape/shape-field
• /NXsnshisto/ENTRY/instrument/CRYSTAL/origin/shape/size-field
• /NXsnshisto/ENTRY/instrument/CRYSTAL/origin/translation-group
• /NXsnshisto/ENTRY/instrument/CRYSTAL/origin/translation/distance-field
• /NXsnshisto/ENTRY/instrument/CRYSTAL/type-field
• /NXsnshisto/ENTRY/instrument/CRYSTAL/wavelength-field
• /NXsnshisto/ENTRY/instrument/DETECTOR-group
• /NXsnshisto/ENTRY/instrument/DETECTOR/azimuthal_angle-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/data-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/data_x_time_of_flight-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/data_x_y-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/data_y_time_of_flight-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/distance-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/origin-group
• /NXsnshisto/ENTRY/instrument/DETECTOR/origin/orientation-group
• /NXsnshisto/ENTRY/instrument/DETECTOR/origin/orientation/value-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/origin/shape-group
• /NXsnshisto/ENTRY/instrument/DETECTOR/origin/shape/description-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/origin/shape/shape-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/origin/shape/size-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/origin/translation-group
• /NXsnshisto/ENTRY/instrument/DETECTOR/origin/translation/distance-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/pixel_id-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/polar_angle-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/time_of_flight-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/total_counts-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/x_pixel_offset-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/y_pixel_offset-field
• /NXsnshisto/ENTRY/instrument/DETECTOR/total_counts-field
• /NXsnshisto/ENTRY/instrument/DISK_CHOPPER-group
• /NXsnshisto/ENTRY/instrument/DISK_CHOPPER/distance-field
• /NXsnshisto/ENTRY/instrument/FERMI_CHOPPER-group
• /NXsnshisto/ENTRY/instrument/FERMI_CHOPPER/distance-field
• /NXsnshisto/ENTRY/instrument/moderator-group
• /NXsnshisto/ENTRY/instrument/moderator/coupling_material-field
• /NXsnshisto/ENTRY/instrument/moderator/distance-field
• /NXsnshisto/ENTRY/instrument/moderator/temperature-field
• /NXsnshisto/ENTRY/instrument/moderator/type-field
• /NXsnshisto/ENTRY/instrument/name-field
• /NXsnshisto/ENTRY/instrument/POLARIZER-group
• /NXsnshisto/ENTRY/instrument/SNS-group
• /NXsnshisto/ENTRY/instrument/SNS/frequency-field
• /NXsnshisto/ENTRY/instrument/SNS/name-field
• /NXsnshisto/ENTRY/instrument/SNS/probe-field
• /NXsnshisto/ENTRY/instrument/SNS/type-field
• /NXsnshisto/ENTRY/instrument/SNSdetector_calibration_id-field
• /NXsnshisto/ENTRY/instrument/SNSgeometry_file_name-field
• /NXsnshisto/ENTRY/instrument/SNStranslation_service-field
• /NXsnshisto/ENTRY/MONITOR-group
• /NXsnshisto/ENTRY/MONITOR/data-field
• /NXsnshisto/ENTRY/MONITOR/distance-field
• /NXsnshisto/ENTRY/MONITOR/mode-field
• /NXsnshisto/ENTRY/MONITOR/time_of_flight-field
• /NXsnshisto/ENTRY/notes-field
• /NXsnshisto/ENTRY/proton_charge-field
• /NXsnshisto/ENTRY/raw_frames-field
• /NXsnshisto/ENTRY/run_number-field
• /NXsnshisto/ENTRY/sample-group
• /NXsnshisto/ENTRY/sample/changer_position-field
• /NXsnshisto/ENTRY/sample/holder-field
• /NXsnshisto/ENTRY/sample/identifier-field
• /NXsnshisto/ENTRY/sample/name-field
• /NXsnshisto/ENTRY/sample/nature-field
• /NXsnshisto/ENTRY/SNSHistoTool-group
• /NXsnshisto/ENTRY/SNSHistoTool/author-field
• /NXsnshisto/ENTRY/SNSHistoTool/command1-field
• /NXsnshisto/ENTRY/SNSHistoTool/date-field
• /NXsnshisto/ENTRY/SNSHistoTool/description-field
• /NXsnshisto/ENTRY/SNSHistoTool/SNSbanking_file_name-field
• /NXsnshisto/ENTRY/SNSHistoTool/SNSmapping_file_name-field
• /NXsnshisto/ENTRY/SNSHistoTool/version-field
• /NXsnshisto/ENTRY/start_time-field
• /NXsnshisto/ENTRY/title-field
• /NXsnshisto/ENTRY/total_counts-field
• /NXsnshisto/ENTRY/total_uncounted_counts-field
• /NXsnshisto/ENTRY/USER-group
• /NXsnshisto/ENTRY/USER/facility_user_id-field
• /NXsnshisto/ENTRY/USER/name-field
• /NXsnshisto/ENTRY/USER/role-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsnshisto.nxdl.xml

NXsolenoid_magnet

Status:
- base class, extends NXobject

Description:
- definition for a solenoid magnet.

Symbols:
- No symbol table

3.3. NeXus Class Definitions
Groups cited:
  
  NXlog

Structure:

  description: (optional) NX_CHAR

  extended description of the magnet.

  beamline_distance: (optional) NX_FLOAT {units=NX_LENGTH}

  define position of beamline element relative to production target

  set_current: (optional) NX_FLOAT {units=NX_CURRENT}

  current set on supply.

  read_current: (optional) NXlog

  current read from supply.

    value: (optional) NX_CHAR {units=NX_CURRENT}

  read_voltage: (optional) NXlog

  voltage read from supply.

    value: (optional) NX_CHAR {units=NX_VOLTAGE}

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

  • /NXsolenoid_magnet/beamline_distance-field
  • /NXsolenoid_magnet/description-field
  • /NXsolenoid_magnet/read_current-group
  • /NXsolenoid_magnet/read_current/value-field
  • /NXsolenoid_magnet/read_voltage-group
  • /NXsolenoid_magnet/read_voltage/value-field
  • /NXsolenoid_magnet/set_current-field

NXDL Source:

  https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsolenoid_magnet.nxdl.xml

NXsolid_geometry

Status:

  base class, extends NXobject

Description:

  the head node for constructively defined geometry

Symbols:

  No symbol table

Groups cited:

  NXcsg, NXoff_geometry, NXquadric
Structure:

**QUADRIC**: (optional) *NXquadric*

Instances of *NXquadric* making up elements of the geometry.

**OFF_GEOMETRY**: (optional) *NXoff_geometry*

Instances of *NXoff_geometry* making up elements of the geometry.

**CSG**: (optional) *NXcsg*

The geometries defined, made up of instances of *NXquadric* and *NXoff_geometry*.

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXsolid_geometry/CSG-group
- /NXsolid_geometry/OFF_GEOMETRY-group
- /NXsolid_geometry/QUADRIC-group

**NXDL Source:**

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsolid_geometry.nxdl.xml

---

**NXspectrum_set_em_auger**

**Status:**

base class, extends *NXobject*

**Description:**

Container for reporting a set of auger electron energy spectra.

**Symbols:**

No symbol table

**Groups cited:**

*NXdata, NXprocess*

**Structure:**

**DATA**: (optional) *NXdata*

**PROCESS**: (optional) *NXprocess*

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXspectrum_set_em_auger/DATA-group
- /NXspectrum_set_em_auger/PROCESS-group

**NXDL Source:**


---

### 3.3. NeXus Class Definitions
**NXspectrum_set_em_cathodolum**

**Status:**
- base class, extends *NXobject*

**Description:**
- Container for reporting a set of cathodoluminescence spectra.

**Symbols:**
- No symbol table

**Groups cited:**
- *NXdata, NXprocess*

**Structure:**
- **DATA:** (optional) *NXdata*
- **PROCESS:** (optional) *NXprocess*

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXspectrum_set_em_cathodolum/DATA-group
- /NXspectrum_set_em_cathodolum/PROCESS-group

**NXDL Source:**

---

**NXspectrum_set_em_eels**

**Status:**
- base class, extends *NXobject*

**Description:**
- Container for reporting a set of electron energy loss spectra.

**Symbols:**
- No symbol table

**Groups cited:**
- *NXdata, NXprocess*

**Structure:**
- **DATA:** (optional) *NXdata*
- **PROCESS:** (optional) *NXprocess*
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXspectrum_set_em_eels/DATA-group
- /NXspectrum_set_em_eels/PROCESS-group

NXDL Source:

NXspectrum_set_em_xray

Status:
base class, extends NXobject

Description:
Container for reporting a set of energy-dispersive X-ray spectra.

Virtually the most important case is that spectra are collected in a scanning microscope (SEM or STEM) for a collection of points. The majority of cases use simple d-dimensional regular scan pattern, such as single point, line profiles, or (rectangular) surface mappings. The latter pattern is the most frequently used.

For now the base class provides for scans where the settings, binning, and energy resolution is the same for each scan point.

IUPAC instead of Siegbahn notation should be used.

Symbols:
- \( n_p \): Number of scan points
- \( n_y \): Number of pixel per Kikuchi pattern in the slow direction
- \( n_x \): Number of pixel per Kikuchi pattern in the fast direction
- \( n_{\text{photon\_energy}} \): Number of X-ray photon energy (bins)
- \( n_{\text{elements}} \): Number of identified elements
- \( n_{\text{peaks}} \): Number of peaks

Groups cited:
NXdata, NXion, NXpeak, NXprocess

Structure:

DATA: (optional) NXdata

Collected X-ray counts chunked based on rectangular images.

This representation supports only rectangular scan pattern.

@long_name: (optional) NX_CHAR

X-ray photon counts

counts: (optional) NX_UINT (Rank: 3, Dimensions: \([n_y, n_x, n_{\text{photon\_energy}}]\))
{units=NX_UNITLESS}

ypos: (optional) NX_NUMBER (Rank: 1, Dimensions: \([n_y]\)) {units=NX_LENGTH}

@long_name: (optional) NX_CHAR
Label for the y axis

xpos: (optional) \textit{NX\_NUMBER} (Rank: 1, Dimensions: [n\_x]) \{units=NX\_LENGTH\}

@\textit{long\_name}: (optional) \textit{NX\_CHAR}

Label for the x axis

photon\_energy: (optional) \textit{NX\_NUMBER} (Rank: 1, Dimensions: [n\_photon\_energy]) \{units=NX\_ENERGY\}

@\textit{long\_name}: (optional) \textit{NX\_CHAR}

X-ray energy

indexing: (optional) \textit{NX\_process}

Details about computational steps how peaks were indexed as elements.

program: (optional) \textit{NX\_CHAR}

Given name of the program that was used to perform this computation.

@\textit{version}: (optional) \textit{NX\_CHAR}

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

element\_names: (optional) \textit{NX\_CHAR} (Rank: 1, Dimensions: [n\_elements])

List of the names of identified elements.

PEAK: (optional) \textit{NX\_peak}

Name and location of each X-ray line which was indexed as a known ion. For each ion an NXion instance should be created which specifies the origin of the signal. For each ion also the relevant IUPAC notation X-ray lines should be specified.

ION: (optional) \textit{NX\_ion}

iupac\_line\_names: (optional) \textit{NX\_CHAR}

IUPAC notation identifier of the line which the peak represents.

This can be a list of IUPAC notations for (the seldom) case that multiple lines are group with the same peak.

composition\_map: (optional) \textit{NX\_process}

Individual element-specific EDX/EDS/EDXS/SXES mapping

A composition map is an image whose intensities for each pixel are the accumulated X-ray quanta \textit{under the curve(s)} of a set of peaks.

program: (optional) \textit{NX\_CHAR}

Given name of the program that was used to perform this computation.

@\textit{version}: (optional) \textit{NX\_CHAR}

Program version plus build number, commit hash, or description of an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured in such a manner that the result file is ideally recreatable yielding the same results.

peaks: (optional) \textit{NX\_CHAR} (Rank: 1, Dimensions: [n\_peaks])
A list of strings of named instances of NXpeak from indexing whose X-ray quanta where accumulated for each pixel.

**name:** (optional) *NX_CHAR*

Human-readable, given name to the image.

**DATA:** (optional) *NXdata*

Individual element-specific maps. Individual maps should each be a group and be named according to element_names.

@long_name: (optional) *NX_CHAR*

Accumulated X-ray photon counts

**counts:** (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_y, n_x])

@long_name: (optional) *NX_CHAR*

Label for the y axis

**xpos:** (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_x])

@long_name: (optional) *NX_CHAR*

Label for the x axis

**Hypertext Anchors**

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXspectrum_set_em_xray/DATA-group
- /NXspectrum_set_em_xray/DATA/counts-field
- /NXspectrum_set_em_xray/DATA/photon_energy-field
- /NXspectrum_set_em_xray/DATA/photon_energy@long_name-attribute
- /NXspectrum_set_em_xray/DATA/xpos-field
- /NXspectrum_set_em_xray/DATA/xpos@long_name-attribute
- /NXspectrum_set_em_xray/DATA/ypos-field
- /NXspectrum_set_em_xray/DATA/ypos@long_name-attribute
- /NXspectrum_set_em_xray/DATA@long_name-attribute
- /NXspectrum_set_em_xray/indexing-group
- /NXspectrum_set_em_xray/indexing/composition_map-group
- /NXspectrum_set_em_xray/indexing/composition_map/DATA-group
- /NXspectrum_set_em_xray/indexing/composition_map/DATA/counts-field
- /NXspectrum_set_em_xray/indexing/composition_map/DATA/xpos-field
- /NXspectrum_set_em_xray/indexing/composition_map/DATA/xpos@long_name-attribute
• /NXspectrum_set_em_xray/indexing/composition_map/DATA/ypos-field
• /NXspectrum_set_em_xray/indexing/composition_map/DATA/ypos@long_name-attribute
• /NXspectrum_set_em_xray/indexing/composition_map/DATA@long_name-attribute
• /NXspectrum_set_em_xray/indexing/composition_map/name-field
• /NXspectrum_set_em_xray/indexing/composition_map/peaks-field
• /NXspectrum_set_em_xray/indexing/composition_map/program-field
• /NXspectrum_set_em_xray/indexing/composition_map/program@version-attribute
• /NXspectrum_set_em_xray/indexing/element_names-field
• /NXspectrum_set_em_xray/indexing/PEAK-group
• /NXspectrum_set_em_xray/indexing/PEAK/ION-group
• /NXspectrum_set_em_xray/indexing/PEAK/ION/iupac_line_names-field
• /NXspectrum_set_em_xray/indexing/program-field
• /NXspectrum_set_em_xray/indexing/program@version-attribute

NXDL Source:

NXspin_rotator

Status:

base class, extends NXobject

Description:

definition for a spin rotator.

Symbols:

No symbol table

Groups cited:

NXlog

Structure:

description: (optional) NX_CHAR

extended description of the spin rotator.

beamline_distance: (optional) NX_FLOAT {units=NX_LENGTH}

define position of beamline element relative to production target

set_Bfield_current: (optional) NX_FLOAT {units=NX_CURRENT}

current set on magnet supply.

set_Efield_voltage: (optional) NX_FLOAT {units=NX_VOLTAGE}

current set on HT supply.

read_Bfield_current: (optional) NXlog
current read from magnet supply.

value: (optional) \texttt{NX\_CHAR} \{units=\texttt{NX\_CURRENT}\}

\textbf{read\_Bfield\_voltage}: (optional) \texttt{NXlog}

voltage read from magnet supply.

value: (optional) \texttt{NX\_CHAR} \{units=\texttt{NX\_VOLTAGE}\}

\textbf{read\_Efield\_current}: (optional) \texttt{NXlog}

current read from HT supply.

value: (optional) \texttt{NX\_CHAR} \{units=\texttt{NX\_CURRENT}\}

\textbf{read\_Efield\_voltage}: (optional) \texttt{NXlog}

voltage read from HT supply.

value: (optional) \texttt{NX\_CHAR} \{units=\texttt{NX\_VOLTAGE}\}

\textbf{Hypertext Anchors}

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXspin\_rotator/beamline\_distance-field
- /NXspin\_rotator/description-field
- /NXspin\_rotator/read\_Bfield\_current-group
- /NXspin\_rotator/read\_Bfield\_current/value-field
- /NXspin\_rotator/read\_Bfield\_voltage-group
- /NXspin\_rotator/read\_Bfield\_voltage/value-field
- /NXspin\_rotator/read\_Efield\_current-group
- /NXspin\_rotator/read\_Efield\_current/value-field
- /NXspin\_rotator/read\_Efield\_voltage-group
- /NXspin\_rotator/read\_Efield\_voltage/value-field
- /NXspin\_rotator/set\_Bfield\_current-field
- /NXspin\_rotator/set\_Efield\_voltage-field

\textbf{NXDL Source:}
https://github.com/nexusformat/definitions/blob/main/contributed definitions/NXspin_rotator.nxdl.xml

\textbf{NXspindispersion}

\textbf{Status:}
base class, extends \texttt{NXobject}

\textbf{Description:}
Subclass of NXelectronanalyser to describe the spin filters in photoemission experiments.

\textbf{Symbols:}
No symbol table
Groups cited:

NXdeflector, Nxlims_em, NXtransformations

Structure:

type: (optional) NX_CHAR

Type of spin detector, VLEED, SPLEED, Mott, etc.

figure_of_merit: (optional) NX_FLOAT {units=NX_DIMENSIONLESS}

Figure of merit of the spin detector

shermann_function: (optional) NX_FLOAT {units=NX_DIMENSIONLESS}

Effective Shermann function, calibrated spin selectivity factor

scattering_energy: (optional) NX_FLOAT {units=NX_ENERGY}

Energy of the spin-selective scattering

scattering_angle: (optional) NX_FLOAT {units=NX_ANGLE}

Angle of the spin-selective scattering

target: (optional) NX_CHAR

Name of the target

target_preparation: (optional) NX_CHAR

Preparation procedure of the spin target

target_preparation_date: (optional) NX_DATE_TIME

Date of last preparation of the spin target

depends_on: (optional) NX_CHAR

Specifies the position of the lens by pointing to the last transformation in the transformation chain in the NXtransformations group.

TRANSFORMATIONS: (optional) NXtransformations

Collection of axis-based translations and rotations to describe the location and geometry of the deflector as a component in the instrument. Conventions from the NXtransformations base class are used. In principle, the McStas coordinate system is used. The first transformation has to point either to another component of the system or . (for pointing to the reference frame) to relate it relative to the experimental setup. Typically, the components of a system should all be related relative to each other and only one component should relate to the reference coordinate system.

DEFLECTOR: (optional) NXdeflector

Deflectors in the spin dispersive section

LENS_EM: (optional) Nxlims_em

Individual lenses in the spin dispersive section
Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXspindispersion/DEFLECTOR-group
- /NXspindispersion/depends_on-field
- /NXspindispersion/figure_of_merit-field
- /NXspindispersion/LENS_EM-group
- /NXspindispersion/scattering_angle-field
- /NXspindispersion/scattering_energy-field
- /NXspindispersion/shermann_function-field
- /NXspindispersion/target-field
- /NXspindispersion/target_preparation-field
- /NXspindispersion/target_preparation_date-field
- /NXspindispersion/TRANSFORMATIONS-group
- /NXspindispersion/type-field

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspindispersion.nxdl.xml

NXstage_lab

Status:
base class, extends NXobject

Description:
A stage lab can be used to hold, align, orient, and prepare a specimen.

Modern stages are multi-functional devices. Many of which offer a controlled environment around (a part) of the specimen. Stages enable experimentalists to apply stimuli. A stage_lab is a multi-purpose-functional tools which can have multiple actuators, sensors, and other components.

With such stages comes the need for storing various (meta)data that are generated while manipulating the sample.

Modern stages realize a hierarchy of components: For example the specimen might be mounted on a multi-axial tilt rotation holder. This holder is fixed in the support unit which connects the holder to the rest of the microscope.

In other examples, taken from atom probe microscopy, researchers may work with wire samples which are clipped into a larger fixing unit for convenience and enable for a more careful specimen handling. This fixture unit is known in atom probe jargon as a stub. Stubs in turn are positioned onto pucks. Pucks are then loaded onto carousels. A carousel is a carrier unit with which eventually entire sets of specimens can be moved in between parts of the microscope.

An NXstage_lab instance reflects this hierarchical design. The stage is the root of the hierarchy. A stage carries the holder. In the case that it is not practical to distinguish these two layers, the holder should be given preference.

Some examples for stage_labs in applications:
• A nanoparticle on a copper grid. The copper grid is the holder. The grid itself is fixed to the stage.
• An atom probe specimen fixed in a stub. In this case the stub can be considered the holder, while the cryostat temperature control unit is a component of the stage.
• Samples with arrays of specimens, like a microtip on a microtip array is an example of a three-layer hierarchy commonly employed for efficient sequential processing of atom probe experiments.
• With one entry of an application definition only one microtip should be described. Therefore, the microtip is the specimen, the array is the holder and the remaining mounting unit that is attached to the cryo-controller is the stage.
• For in-situ experiments with e.g. chips with read-out electronics as actuators, the chips are again placed in a larger unit.
• Other examples are (quasi) in-situ experiments where experimentalists anneal or deform the specimen via e.g. in-situ tensile testing machines which are mounted on the specimen holder.

To cover for an as flexible design of complex stages, users should nest multiple instances of NXstage_lab objects according to their needs to reflect the differences between what they consider as the holder and what they consider is the stage.

Instances should be named with integers starting from 1 as the top level unit. In the microtip example stage_lab_1 for the stage, stage_lab_2 for the holder (microtip array), stage_lab_3 for the microtip specimen, respectively. The depends_on keyword should be used with relative or absolute naming inside the file to specify how different stage_lab instances build a hierarchy if this is not obvious from numbered identifiers like the stage_lab_1 to stage_lab_3 example. The lower it is the number the higher it is the rank in the hierarchy.

For specific details and inspiration about stages in electron microscopes:
• Holders with multiple axes
• Chip-based designs
• Further chip-based designs
• Stages in transmission electron microscopy (page 103, table 4.2)
• Further stages in transmission electron microscopy (page 124ff)
• Specimens in atom probe (page 47ff)
• Exemplar micro-manipulators

Symbols:
No symbol table

Groups cited:
NXmanufacturer, NXpositioner, NXtransformations

Structure:

design: (optional) NX_CHAR
Principal design of the stage.
Exemplar terms could be side_entry, top_entry, single_tilt, quick_change, multiple_specimen, bulk_specimen, double_tilt, tilt_rotate, heating_chip, atmosphere_chip, electrical_biasing_chip, liquid_cell_chip

name: (optional) NX_CHAR
Given name/alias for the components making the stage.
description: (optional) NX_CHAR

Ideally, a (globally) unique persistent identifier, link, or text to a resource which gives further details.

tilt_1: (optional) NX_FLOAT {units=NX_ANGLE}

Should be defined by the application definition.

tilt_2: (optional) NX_FLOAT {units=NX_ANGLE}

Should be defined by the application definition.

rotation: (optional) NX_FLOAT {units=NX_ANGLE}

Should be defined by the application definition.

position: (optional) NX_FLOAT (Rank: 1, Dimensions: [3]) {units=NX_LENGTH}

Should be defined by the application definition.

bias_voltage: (optional) NX_FLOAT {units=NX_VOLTAGE}

Voltage applied to the stage to decelerate electrons.

MANUFACTURER: (optional) NXmanufacturer

TRANSFORMATIONS: (optional) NXtransformations

The rotation, tilt and position of stage components can be specified either via NXtransformations or via the tilt_1, tilt_2, rotation, and position fields.

POSITIONER: (optional) NXpositioner

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXstage_lab/bias_voltage-field
- /NXstage_lab/description-field
- /NXstage_lab/design-field
- /NXstage_lab/MANUFACTURER-group
- /NXstage_lab/name-field
- /NXstage_lab/position-field
- /NXstage_lab/POSITIONER-group
- /NXstage_lab/rotation-field
- /NXstage_lab/tilt_1-field
- /NXstage_lab/tilt_2-field
- /NXstage_lab/TRANSFORMATIONS-group

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXstage_lab.nxdl.xml
NXxpcs

Status:

application definition, extends NXobject

Description:

X-ray Photon Correlation Spectroscopy (XPCS) data (results).

The purpose of NXxpcs is to document and communicate an accepted vernacular for various XPCS results data in order to support development of community software tools. The definition presented here only represents a starting point and contains fields that a common software tool should support for community acceptance.

Additional fields may be added to XPCS results file (either formally or informally). It is expected that this XPCS data will be part of multi-modal data set that could involve e.g., NXcanSAS or 1D and/or 2D data.

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:

NXbeam, NXdata, NXdetector, NXentry, NXinstrument, NXnote, NXpositioner, NXprocess, NXsample

Structure:

entry: (required) NXentry

    definition: (required) NX_CHAR

        Official NeXus NXDL schema to which this file conforms

        Obligatory value: NXxpcs

    entry_identifier: (required) NX_CHAR

        Locally unique identifier for the experiment (a.k.a. run or scan).

        • For bluesky users, this is the run’s “scan_id”.

        • For SPEC users, this is the scan number (SCAN_N).

    entry_identifier_uuid: (optional) NX_CHAR

        (optional) UUID identifier for this entry.

        See the UUID standard (or wikipedia) for more information.

        • For bluesky users, this is the run’s “uid” and is expected for that application.

        • Typically, SPEC users will not use this field without further engineering.

    scan_number: (required) NX_INT

        DEPRECATED: Use the entry_identifier field.

        Scan number (must be an integer).

        NOTE: Link to collection_identifier.

    start_time: (required) NX_DATE_TIME

        Starting time of experiment, such as “2021-02-11 11:22:33.445566Z”.

    end_time: (optional) NX_DATE_TIME


Ending time of experiment, such as “2021-02-11 11:23:45Z”.

**data**: (required) **NXdata**

The results data captured here are most commonly required for high throughput, equilibrium dynamics experiments. Data (results) describing on-equilibrium dynamics consume more memory resources so these data are separated.

**frame_sum**: (optional) **NX_NUMBER** {units=NX_COUNT}

Two-dimensional summation along the frames stack.

sum of intensity v. time (in the units of “frames”)

**frame_average**: (optional) **NX_NUMBER** {units=NX_COUNT}

Two-dimensional average along the frames stack.

average intensity v. time (in the units of “frames”)

**g2**: (optional) **NX_NUMBER** {units=NX_DIMENSIONLESS}

normalized intensity auto-correlation function, see Lumma, Rev. Sci. Instr. (2000), Eq 1

\[ g_2(Q, t) = \frac{\langle I(Q, t)I(Q, t + t) \rangle}{\langle I(Q, t) \rangle^2}; t > 0 \]

Typically, \( g_2 \) is a quantity calculated for a group of pixels representing a specific region of reciprocal space. These groupings, or bins, are generically described as \( q \). Some open-source XPCS libraries refer to these bins as “rois”, which are not to be confused with EPICS AreaDetector ROI. See usage guidelines for \( q \_lists \) and \( roi \_maps \) within a mask.\(^1\)

In short, \( g_2 \) should be ordered according to the \( roi \_map \) value. In principle, any format is acceptable if the data and its axes are self-describing as per NeXus recommendations. However, the data is preferred in one of the following two formats:

- iterable list of linked files (or keys) for each \( g_2 \) with 1 file (key) per \( q \), where \( q \) is called by the nth \( roi \_map \) value
- 2D array\(^2\) with shape \((g_2, q)\), where \( q \) is represented by the nth \( roi \_map \) value, not the value \( q \) value

Note it is expected that “g2” and all quantities following it will be of the same length.

Other formats are acceptable with sufficient axes description.

See references below for related implementation information:

**@storage_mode**: (required) **NX_CHAR**

storage_mode describes the format of the data to be loaded

We encourage the documentation of other formats not represented here.

- one array representing entire data set (“one_array”)
- data exchange format with each key representing one \( q \) by its corresponding \( roi \_map \) value (“data_exchange_keys”)

---

\(^1\) mask: `NXpcs:/entry/instrument/masks-group`

\(^2\) NeXus 2-D data and axes: [https://manual.nexusformat.org/classes/base_classes/NXdata.html#nxdata](https://manual.nexusformat.org/classes/base_classes/NXdata.html#nxdata)
Any of these values: one_array | data_exchange_keys | other

**g2_der**: (optional) `NX_NUMBER` \{units=NX\_DIMENSIONLESS\}

error values for the $g_2$ values.

The derivation of the error is left up to the implemented code. Symmetric error will be expected ($\pm$ error). The data should be in the same format as $g_2$.

@**storage_mode**: (required) `NX_CHAR`

Any of these values: one_array | data_exchange_keys | other

**G2\_unnormalized**: (optional) `NX_NUMBER` \{units=NX\_DIMENSIONLESS\}

unnormalized intensity auto-correlation function.

Specifically, $g_2$ without the denominator. The data should be in the same format as $g_2$.

@**storage_mode**: (required) `NX_CHAR`

Any of these values: one_array | data_exchange_keys | other

**delay\_difference**: (optional) `NX\_INT` \{units=NX\_INT\}

delay\_difference (also known as delay or lag step)

This is quantized difference so that the “step” between two consecutive frames is one frame (or step $dt = 1$ frame)

It is the “quantized” delay time corresponding to the $g_2$ values.

The unit of delay\_differences is NX\_INT for units of frames (i.e., integers) preferred, refer to NXdetector for conversion to time units.

@**storage_mode**: (required) `NX_CHAR`

Any of these values: one_array | data_exchange_keys | other

**twotime**: (optional) `NXdata`

The data (results) in this section are based on the two-time intensity correlation function derived from a time series of scattering images.

**two\_time\_corr\_func**: (optional) `NX_NUMBER` \{units=NX\_ANY\}

two-time correlation of speckle intensity for a given q-bin or roi (represented by the nth roi\_map value)

See Fluerasu, Phys Rev E (2007), Eq 1 and Sutton, Optics Express (2003) for an early description applied to X-ray scattering:

$$C(Q,t_1,t_2) = \frac{\langle I(Q,t_1) I(Q,t_2) \rangle}{\langle I(Q,t_1) \rangle \langle I(Q,t_2) \rangle}$$

in which time is quantized by frames. In principle, any data format is acceptable if the data and its axes are self-describing as per NeXus recommendations. However, the data is preferred in one of the following two formats:

- iterable list of linked files (or keys) for each q-bin called by the nth roi\_map value. data for each bin is a 2D array
- 3D array with shape (frames, frames, q) or (q, frames, frames), where q is represented by the nth roi\_map value, not the value q value
The computation of this result can be customized. These customizations can affect subsequently derived results (below). The following attributes will be used to manage the customization.

- Other normalization methods may be applied, but the method will not be specified in this definition. Some of these normalization methods result in a baseline value of 0, not 1.
- The various software libraries use different programming languages. Therefore, we need to specify the \( \text{time} = 0 \) origin location of the 2D array for each \( q \).
- A method to reduce data storage needs is to only record half of the 2D array by populating array elements above or below the array diagonal.

@\textbf{storage\_mode}: (required) \textit{NX\_CHAR}

\texttt{storage\_mode} describes the format of the data to be loaded

We encourage the documentation of other formats represented here.

Any of these values:
- \texttt{one\_array\_q\_first}
- \texttt{one\_array\_q\_last}
- \texttt{data\_exchange\_keys}
- \texttt{other}

@\textbf{baseline\_reference}: (required) \textit{NX\_INT}

\texttt{baseline} is the expected value of a full decorrelation

The baseline is a constant value added to the functional form of the autocorrelation function. This value is required.

Any of these values: \( 0 \mid 1 \)

@\textbf{time\_origin\_location}: (required) \textit{NX\_CHAR}

\texttt{time\_origin\_location} is the location of the origin

Any of these values: \texttt{upper\_left} | \texttt{lower\_left}

@\textbf{populated\_elements}: (required) \textit{NX\_CHAR}

\texttt{populated\_elements} describe the elements of the 2D array that are populated with data

Any of these values: \texttt{all} | \texttt{upper\_half} | \texttt{lower\_half}

\textbf{\texttt{g2\_from\_two\_time\_corr\_func}: (optional) \textit{NX\_NUMBER}}

\texttt{\{units=NX\_DIMENSIONLESS\}}

frame weighted average along the diagonal direction in \texttt{two\_time\_corr\_func}

The data format and description should be consistent with that found in “/NXxpcs/entry/data/g2”

- iterative list of linked files for each \( g_2 \) with 1 file per \( q \)
- 2D array with shape \((g_2, q)\)
Note that delay_difference is not included here because it is derived from the shape of extracted $g_2$ because all frames are considered, which is not necessarily the case for $g_2$.

The computation of this result can be customized. The customization can affect the fitting required to extract quantitative results. The following attributes will be used to manage the customization.

@storage_mode: (required) *NX_CHAR*

Any of these values:

- one_array_q_first
- one_array_q_last
- data_exchange_keys
- other

@baseline_reference: (required) *NX_INT*

Any of these values: 0 | 1

@first_point_for_fit: (required) *NX_INT*

first_point_for_fit describes if the first point should or should not be used in fitting the functional form of the dynamics to extract quantitative time-scale information.

The first_point_for_fit is True (“1”) or False (“0”). This value is required.

Any of these values: 0 | 1

$g_2\text{err from two_time_corr_func}$: (optional) *NX_NUMBER*

{units=*NX_DIMENSIONLESS*}

error values for the $g_2$ values.

The derivation of the error is left up to the implemented code. Symmetric error will be expected ($\pm$ error).

@storage_mode: (required) *NX_CHAR*

Any of these values:

- one_array_q_first
- one_array_q_last
- data_exchange_keys
- other

$g_2\text{from two_time_corr_func partials}$: (optional) *NX_NUMBER*

{units=*NX_DIMENSIONLESS*}

subset of frame weighted average along the diagonal direction in two_time_corr_func

Time slicing along the diagonal can be very sophisticated. This entry currently assumes equal frame-binning. The data formats are highly dependent on the implantation of various analysis libraries. In principle, any data format is acceptable if the data and its axes are self describing as per NeXus
recommendations. However, the data is preferred in one of the following two formats:

- iterable list of linked files (or keys) for each partial $g_2$ of each q-bin represented by the roi_map value
- 3D array with shape ($g_2$, $q$, nth_partial)

Note that delay_difference is not included here because it is derived from the shape of extracted $g_2$.

@storage_mode: (required) NX_CHAR

Any of these values: one_array | data_exchange_keys | other

@baseline_reference: (required) NX_INT

Any of these values: 0 | 1

g2_err_from_two_time_corr_func_partials: (optional) NX_NUMBER {units=NX_DIMENSIONLESS}

error values for the $g_2$ values.

The derivation of the error is left up to the implemented code. Symmetric error will be expected ($\pm$ error).

instrument: (required) NXinstrument

XPCS instrument Metadata.

Objects can be entered here directly or linked from other objects in the NeXus file (such as within /entry/instrument).

incident_beam: (required) NXbeam

incident_energy: (required) NX_FLOAT {units=NX_ENERGY}

Incident beam line energy (either keV or eV).

incident_energy_spread: (optional) NX_FLOAT {units=NX_ENERGY}

Spread of incident beam line energy (either keV or eV). This quantity is otherwise known as the energy resolution, which is related to the longitudinal coherence length.

incident_polarization_type: (optional) NX_CHAR

Terse description of the incident beam polarization.

The value can be plain text, such as vertical, C+, circular left.

extent: (optional) NX_FLOAT {units=NX_LENGTH}

Size (2-D) of the beam at this position.

DETECTOR: (required) NXdetector

XPCS data is typically produced by area detector (likely EPICS AreaDetector) as a stack of 2D images. Sometimes this data is represented in different ways (sparse arrays or photon event list), but this detail is left to the analysis software. Therefore, we only include requirements based on full array data.

We note that the image origin (pixel coordinates (0,0)) are found at the top left of a single 2D image array. This is the standard expected by Coherent X-ray
Imaging Data Bank.\(^3\) See CXI version 1.6 and Maia, Nature Methods (2012). This seems to be consistent with matplotlib and the practiced implementation of EPICS AreaDetector. However, some exceptions may exist in the CXI documentation (See Fig 11 vs Fig 12).

Additionally, not all \texttt{NXdetector} dependencies are inherited from AreaDetector or other control systems. \texttt{frame_time} is used to convert \texttt{delay_difference} to seconds. \texttt{frame_time} field could be missing from AreaDetector or may either be \texttt{acquire_period} or \texttt{acquire_time}, depending on the detector model and the local implementation.

description: (optional) \texttt{NX_CHAR}

Detector name.

distance: (optional) \texttt{NX_NUMBER \{units=NX_LENGTH\}}

Distance between sample and detector.

count_time: (required) \texttt{NX_NUMBER \{units=NX_TIME\}}

Exposure time of frames, s.

frame_time: (required) \texttt{NX_NUMBER \{units=NX_TIME\}}

Exposure period (time between frame starts) of frames, s

beam_center_x: (required) \texttt{NX_NUMBER \{units=NX_LENGTH\}}

Position of beam center, x axis, in detector’s coordinates.

beam_center_y: (required) \texttt{NX_NUMBER \{units=NX_LENGTH\}}

Position of beam center, y axis, in detector’s coordinates.

x_pixel_size: (optional) \texttt{NX_NUMBER \{units=NX_LENGTH\}}

Length of pixel in x direction.

y_pixel_size: (optional) \texttt{NX_NUMBER \{units=NX_LENGTH\}}

Length of pixel in y direction.

masks: (optional) \texttt{NXnote}

Data masks or mappings to regions of interest (roi) for specific \(Q\) values

Fields in this \texttt{masks} group describe regions of interest in the data by either a mask to select pixels or to associate a \texttt{map} of rois with a (one-dimensional) \texttt{list} of values.

“roi\_maps” provide for representation of pixel binning that are arbitrary and irregular, which is geometry scattering agnostic and most flexible. The maps work as a labeled array for N rois.

“Dynamic” represents quantities directly related to XPCS and NXxpcs\texttt{/entry/data} and NXxpcs\texttt{/entry/two\_time}.

“Static” refers to finer binning used for computation not strictly used for the final XPCS results. Implementation of \_static\_ binning is left for individual libraries to document. We encourage usage of \texttt{NXcanSAS} to represent standard SAXS results or development of new NeXus definitions for GI-SAXS or other reciprocal space intensity mapping.

\(^3\) Coherent X-ray Imaging Data Bank: https://cxidb.org/cxi.html
**dynamic_roi_map**: (required) **NX_NUMBER**

    {units=NX_DIMENSIONLESS}

roi index array or labeled array

The values of this mask index (or map to) the $Q$ value from the the `dynamic_q_list` field. Not that the value of 0 represents in-action. XPCS computations are performed on all pixels with a value > 0.

The `units` attribute should be set to "au" indicating arbitrary units.

**dynamic_q_list**: (optional) **NX_NUMBER** {units=NX_PER_LENGTH}

1-D list of $Q$ values, one for each roi index value.

List order is determined by the index value of the associated roi map starting at 1.

The only requirement for the list is that it may be iterable. Some expected formats are:

- iterable list of floats (i.e., $Q(r)$)
- iterable list of tuples (i.e., $Q(r)$, $\varphi$), but preferable use the separate $\varphi$ field below
- iterable list of tuples (e.g., (H, K, L); (qx, qy, qz); (horizontal_pixel, vertical_pixel))
- iterable list of integers (for Nth roi_map value) or strings

This format is chosen because results plotting packages are not common and simple I/O is required by end user. The lists can be accessed as lists, arrays or via keys

**dynamic_phi_list**: (optional) **NX_NUMBER** {units=NX_PER_LENGTH}

Array of $\varphi$ value for each pixel.

List order is determined by the index value of the associated roi map starting at 1.

**static_roi_map**: (optional) **NX_NUMBER** {units=NX_DIMENSIONLESS}

roi index array.

The values of this mask index the $|Q|$ value from the the `static_q_list` field.

The `units` attribute should be set to "au" indicating arbitrary units.

**static_q_list**: (optional) **NX_NUMBER** {units=NX_PER_LENGTH}

1-D list of $|Q|$ values, 1 for each roi.

**sample**: (optional) **NXSample**

**temperature_set**: (optional) **NX_NUMBER** {units=NX_TEMPERATURE}

Sample temperature setpoint, (C or K).

**temperature**: (optional) **NX_NUMBER** {units=NX_TEMPERATURE}

Sample temperature actual, (C or K).
position_x: (optional) NXpositioner
position_y: (optional) NXpositioner
position_z: (optional) NXpositioner

NOTE: (optional) N NXnote
Any other notes.

NAME: The NeXus convention, to use all upper case to indicate the name (here NOTE), is left to the file writer. In our case, follow the suggested name pattern and sequence: note_1, note_2, note_3, ... Start with note_1 if the first one, otherwise pick the next number in this sequence.

PROCESS: (required) NXprocess
Describe the computation process that produced these results.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- /NXpcs/entry-group
- /NXpcs/entry/data-group
- /NXpcs/entry/data/delay_difference-field
- /NXpcs/entry/data/delay_difference@storage_mode-attribute
- /NXpcs/entry/data/frame_average-field
- /NXpcs/entry/data/frame_sum-field
- /NXpcs/entry/data/g2-field
- /NXpcs/entry/data/g2@storage_mode-attribute
- /NXpcs/entry/data/g2_derr-field
- /NXpcs/entry/data/g2_derr@storage_mode-attribute
- /NXpcs/entry/data/G2_unnormalized-field
- /NXpcs/entry/data/G2_unnormalized@storage_mode-attribute
- /NXpcs/entry/definition-field
- /NXpcs/entry/end_time-field
- /NXpcs/entry/entry_identifier-field
- /NXpcs/entry/entry_identifier_uuid-field
- /NXpcs/entry/instrument-group
- /NXpcs/entry/instrument/DETECTOR-group
- /NXpcs/entry/instrument/DETECTOR/beam_center_x-field
- /NXpcs/entry/instrument/DETECTOR/beam_center_y-field
- /NXpcs/entry/instrument/DETECTOR/count_time-field
- /NXpcs/entry/instrument/DETECTOR/description-field
- /NXpcs/entry/instrument/DETECTOR/distance-field
• /NXpcs/entry/instrument/DETECTOR/frame_time-field
• /NXpcs/entry/instrument/DETECTOR/x_pixel_size-field
• /NXpcs/entry/instrument/DETECTOR/y_pixel_size-field
• /NXpcs/entry/instrument/incident_beam-group
• /NXpcs/entry/instrument/incident_beam/extent-field
• /NXpcs/entry/instrument/incident_beam/incident_energy-field
• /NXpcs/entry/instrument/incident_beam/incident_energy_spread-field
• /NXpcs/entry/instrument/incident_beam/incident_polarization_type-field
• /NXpcs/entry/instrument/masks-group
• /NXpcs/entry/instrument/masks/dynamic_phi_list-field
• /NXpcs/entry/instrument/masks/dynamic_q_list-field
• /NXpcs/entry/instrument/masks/dynamic_roi_map-field
• /NXpcs/entry/instrument/masks/static_q_list-field
• /NXpcs/entry/instrument/masks/static_roi_map-field
• /NXpcs/entry/NOTE-group
• /NXpcs/entry/sample-group
• /NXpcs/entry/sample/position_x-group
• /NXpcs/entry/sample/position_y-group
• /NXpcs/entry/sample/position_z-group
• /NXpcs/entry/sample/temperature-field
• /NXpcs/entry/sample/temperature_set-field
• /NXpcs/entry/start_time-field
• /NXpcs/entry/twotime-group
• /NXpcs/entry/twotime/g2_err_from_two_time_corr_func-field
• /NXpcs/entry/twotime/g2_err_from_two_time_corr_func@storage_mode-attribute
• /NXpcs/entry/twotime/g2_err_from_two_time_corr_funcpartials-field
• /NXpcs/entry/twotime/g2_from_two_time_corr_func-field
• /NXpcs/entry/twotime/g2_from_two_time_corr_func@baseline_reference-attribute
• /NXpcs/entry/twotime/g2_from_two_time_corr_func@first_point_for_fit-attribute
• /NXpcs/entry/twotime/g2_from_two_time_corr_func@storage_mode-attribute
• /NXpcs/entry/twotime/g2_from_two_time_corr_funcpartials-field
• /NXpcs/entry/twotime/g2_from_two_time_corr_funcpartials@baseline_reference-attribute
• /NXpcs/entry/twotime/g2_from_two_time_corr_funcpartials@storage_mode-attribute
• /NXpcs/entry/twotime/two_time_corr_func-field
• /NXpcs/entry/twotime/two_time_corr_func@baseline_reference-attribute
• /NXxpcs/entry/twotime/two_time_corr_func@populated_elements-attribute
• /NXxpcs/entry/twotime/two_time_corr_func@storage_mode-attribute
• /NXxpcs/entry/twotime/two_time_corr_func@time_origin_location-attribute
• /NXxpcs/PROCESS-group

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXxpcs.nxdl.xml

### 3.3.4 Downloads

See this table for the different formats available:

<table>
<thead>
<tr>
<th>download file</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>_static/nxdl_vocabulary.html</code></td>
<td>Human-readable HTML list of anchors, by vocabulary term, with links to the manual.</td>
</tr>
<tr>
<td><code>_static/nxdl_vocabulary.json</code></td>
<td>vocabulary list by key in JSON format</td>
</tr>
<tr>
<td><code>_static/nxdl_vocabulary.txt</code></td>
<td>list of all anchors, sorted alphabetically</td>
</tr>
<tr>
<td><code>_static/nxdl_vocabulary.yml</code></td>
<td>vocabulary list by key in YAML format</td>
</tr>
</tbody>
</table>

---

2 JSON: https://www.w3schools.com/whatis/whatis_json.asp
3 YAML https://yaml.org


NAPI: NEXUS APPLICATION PROGRAMMER INTERFACE (FROZEN)

4.1 Status

This application program interface (API) was developed to support the reading and writing of NeXus files through unified function calls, regardless of the physical data format (XML, HDF4, HDF5).

In the meantime it has been decided that active development of NeXus definitions and tools will concentrate on HDF5 as the only supported physical data format. It is expected that most application developers will use standard HDF5 tools to read and write NeXus. Two examples are provided in *HDF5 in C with libhdf5*.

Therefore, the decision has been taken to freeze the NAPI. Maintenance is reduced to bug fixes.

4.2 Overview

The core routines have been written in C but wrappers are available for a number of other languages including C++, Fortran 77, Fortran 90, Java, Python and IDL. The API makes the reading and writing of NeXus files transparent; the user doesn’t even need to know the underlying format when reading a file since the API calls are the same.

The NeXus Application Programming Interface for the various language backends is available on-line from https://github.com/nexusformat/code/

The NeXusIntern.pdf document (https://github.com/nexusformat/code/blob/master/doc/api/NeXusIntern.pdf) describes the internal workings of the NeXus-API. You are very welcome to read it, but it will not be of much use if all you want is to read and write files using the NAPI.

The NeXus Application Program Interface call routines in the appropriate backend (HDF4, HDF5 or XML) to read and write files with the correct structure. The API serves a number of purposes:

1. It simplifies the reading and writing of NeXus files.

2. It ensures a certain degree of compliance with the NeXus standard.

3. It hides the implementation details of the format. In particular, the API can read and write HDF4, HDF5, and XML files using the same routines.
4.3 Core API

The core API provides the basic routines for reading, writing and navigating NeXus files. Operations are performed using a handle that keeps a record of its current position in the file hierarchy. All are read or write requests are then implicitly performed on the currently open entity. This limits number of parameters that need to be passed to API calls, at the cost of forcing a certain mode of operation. It is very similar to navigating a directory hierarchy; NeXus groups are the directories, which can contain data sets and/or other directories.

The core API comprises the following functional groups:

- General initialization and shutdown: opening and closing the file, creating or opening an existing group or dataset, and closing them.
- Reading and writing data and attributes to previously opened datasets.
- Routines to obtain meta-data and to iterate over component datasets and attributes.
- Handling of linking and group hierarchy.
- Routines to handle memory allocation. (Not required in all language bindings.)

4.3.1 NAPI C and C++ Interface

Documentation is provided online:

C

C++

4.3.2 NAPI Fortran 77 Interface

The bindings are listed at https://github.com/nexusformat/code/tree/master/bindings/f77 and can be built as part of the API distribution https://github.com/nexusformat/code/releases

4.3.3 NAPI Fortran 90 Interface

The Fortran 90 interface is a wrapper to the C interface with nearly identical routine definitions. As with the Fortran 77 interface, it is necessary to reverse the order of indices in multidimensional arrays, compared to an equivalent C program, so that data are stored in the same order in the NeXus file.

Any program using the F90 API needs to put the following line at the top (after the PROGRAM statement):

```
use NXmodule
```

Use the following table to convert from the C data types listed with each routine to the Fortran 90 data types.
<table>
<thead>
<tr>
<th>C data type</th>
<th>F90 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>int, int</td>
<td>integer</td>
</tr>
<tr>
<td>char*</td>
<td>character(len=*)</td>
</tr>
<tr>
<td>NXhandle, NXhandle*</td>
<td>type(NXhandle)</td>
</tr>
<tr>
<td>NXstatus</td>
<td>integer</td>
</tr>
<tr>
<td>int[]</td>
<td>integer(:)</td>
</tr>
<tr>
<td>void*</td>
<td>real(:) or integer(:) or character(len=*)</td>
</tr>
<tr>
<td>NXlink a,NXlink*</td>
<td>type(NXlink)</td>
</tr>
</tbody>
</table>

The parameters in the next table, defined in **NXmodule**, may be used in defining variables.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NX_MAXRANK</td>
<td>Maximum number of dimensions</td>
<td>32</td>
</tr>
<tr>
<td>NX_MAXNAMELEN</td>
<td>Maximum length of NeXus name</td>
<td>64</td>
</tr>
<tr>
<td>NXi1</td>
<td>Kind parameter for a 1-byte integer</td>
<td>selected_int_kind(2)</td>
</tr>
<tr>
<td>NXi2</td>
<td>Kind parameter for a 2-byte integer</td>
<td>selected_int_kind(4)</td>
</tr>
<tr>
<td>NXi4</td>
<td>Kind parameter for a 4-byte integer</td>
<td>selected_int_kind(8)</td>
</tr>
<tr>
<td>NXr4</td>
<td>Kind parameter for a 4-byte real</td>
<td>kind(1.0)</td>
</tr>
<tr>
<td>NXr8</td>
<td>Kind parameter for an 8-byte real</td>
<td>kind(1.0D0)</td>
</tr>
</tbody>
</table>

The bindings are listed at [https://github.com/nexusformat/code/tree/master/bindings/f90](https://github.com/nexusformat/code/tree/master/bindings/f90) and can be built as part of the API distribution [https://github.com/nexusformat/code/releases](https://github.com/nexusformat/code/releases).

### 4.3.4 NAPI Java Interface

This section includes installation notes, instructions for running NeXus for Java programs and a brief introduction to the API.

The Java API for NeXus (**jnexus**) was implemented through the Java Native Interface (JNI) to call on to the native C library. This has a number of disadvantages over using pure Java, however the most popular file backend HDF5 is only available using a JNI wrapper anyway.

**Acknowledgement**

This implementation uses classes and native methods from NCSA’s Java HDF Interface project. Basically all conversions from native types to Java types is done through code from the NCSA HDF group. Without this code the implementation of this API would have taken much longer. See NCSA’s copyright for more information.

**Installation**

**Requirements**

**Caution:** Documentation is old and may need revision.

For running an application with **jnexus** an recent Java runtime environment (JRE) will do.

In order to compile the Java API for NeXus a Java Development Kit is required on top of the build requirements for the C API.

4.3. Core API
**Installation under Windows**

1. Copy the HDF DLL's and the file jnexus.dll to a directory in your path. For instance `C:\Windows\system32`.
2. Copy the `jnexus.jar` to the place where you usually keep library jar files.

Note that the location or the naming of these files in the binary Nexus distributions have changed over the years. In the Nexus 4.3.0 Windows 64-bit distribution (see Assets in [https://github.com/nexusformat/code/releases/tag/4.3.0](https://github.com/nexusformat/code/releases/tag/4.3.0)), by default, the DLL is at: `C:\Program Files\NeXus Data Format\bin\libjnexus-0.dll`. Please rename this file to `jnexus.dll` before making it available in your path. This is important, otherwise, JVM runtime will not be able to locate this file.

For the same distribution, the location of `jnexus.jar` is at: `C:\Program Files\NeXus Data Format\share\java`.

**Installation under Unix**

The `jnexus.so` shared library as well as all required file backend .so libraries are required as well as the `jnexus.jar` file holding the required Java classes. Copy them wherever you like and see below for instructions how to run programs using jnexus.

**Running Programs with the NeXus API for Java**

In order to successfully run a program with jnexus, the Java runtime systems needs to locate two items:

1. The shared library implementing the native methods.
2. The `nexus.jar` file in order to find the Java classes.

**Locating the shared libraries**

The methods for locating a shared library differ between systems. Under Windows32 systems the best method is to copy the `jnexus.dll` and the HDF4, HDF5 and/or XML-library DLL files into a directory in your path.

On a UNIX system, the problem can be solved in three different ways:

1. Make your system administrator copy the `jnexus.so` file into the systems default shared library directory (usually `/usr/lib` or `/usr/local/lib`).
2. Put the `jnexus.so` file wherever you see fit and set the `LD_LIBRARY_PATH` environment variable to point to the directory of your choice.
3. Specify the full pathname of the jnexus shared library on the java command line with the `-Dorg.nexusformat.JNEXUSLIB=full-path-2-shared-library` option.
**Locating jnexus.jar**

This is easier, just add the the full pathname to jnexus.jar to the classpath when starting java. Here are examples for a UNIX shell and the Windows shell.

**UNIX example shell script to start jnexus.jar**

```bash
#!/sbin/sh
java -classpath /usr/lib/classes.zip:../jnexus.jar:. 
-Dorg.nexusformat.JNEXUSLIB=../libjnexus.so TestJapi
```

**Windows 32 example batch file to start jnexus.jar**

```bash
set JL=-Dorg.nexusformat.JNEXUSLIB=../jnexus\bin\win32\jnexus.dll
java -classpath C:\jdk1.5\lib\classes.zip;..\jnexus.jar;. %JL% TestJapi
```

**Programming with the NeXus API for Java**

The NeXus C-API is good enough but for Java a few adaptions of the API have been made in order to match the API better to the idioms used by Java programmers. In order to understand the Java-API, it is useful to study the NeXus C-API because many methods work in the same way as their C equivalents. A full API documentation is available in Java documentation format. For full reference look especially at:

- The interface NeXusFileInterface first. It gives an uncluttered view of the API.
- The implementation NexusFile which gives more details about constructors and constants. However this documentation is interspersed with information about native methods which should not be called by an application programmer as they are not part of the standard and might change in future.

See the following code example for opening a file, opening a vGroup and closing the file again in order to get a feeling for the API:

**fragment for opening and closing**

```java
try{
    NexusFile nf = new NexusFile(filename, NexusFile.NXACC_READ);
    nf.opengroup("entry1","NXentry");
    nf.finalize();
} catch(NexusException ne) {
    // Something was wrong!
}
```

Some notes on this little example:

- Each NeXus file is represented by a NexusFile object which is created through the constructor.
- The NexusFile object takes care of all file handles for you. So there is no need to pass in a handle anymore to each method as in the C language API.
- All error handling is done through the Java exception handling mechanism. This saves all the code checking return values in the C language API. Most API functions return void.
• Closing files is tricky. The Java garbage collector is supposed to call the finalize method for each object it decides to delete. In order to enable this mechanism, the NXclose function was replaced by the finalize method. In practice it seems not to be guaranteed that the garbage collector calls the finalize method. It is safer to call finalize yourself in order to properly close a file. Multiple calls to the finalize method for the same object are safe and do no harm.

Data Writing and Reading

Again a code sample which shows how this looks like:

fragment for writing and reading

```java
int idata[][] = new idata[10][20];
int iDim[] = new int[2];

// put some data into idata....... 

// write idata
iDim[0] = 10;
iDim[1] = 20;
f.makedata("idata",NexusFile.NX_INT32,2,iDim);
f.opendata("idata");
f.putdata(idata);

// read idata
f.getdata(idata);
```

The dataset is created as usual with makedata and opened with putdata. The trick is in putdata. Java is meant to be type safe. One would think then that a putdata method would be required for each Java data type. In order to avoid this, the data to write is passed into putdata as type Object. Then the API proceeds to analyze this object through the Java introspection API and convert the data to a byte stream for writing through the native method call. This is an elegant solution with one drawback: An array is needed at all times. Even if only a single data value is written (or read) an array of length one and an appropriate type is the required argument.

Another issue are strings. Strings are first class objects in Java. HDF (and NeXus) sees them as dumb arrays of bytes. Thus strings have to be converted to and from bytes when reading string data. See a writing example:

String writing

```java
String ame = "Alle meine Entchen";
f.makedata("string_data",NexusFile.NX_CHAR, 
1,ame.length()+2);
f.opendata("string_data");
f.putdata(ame.getBytes());
```

And reading:
String reading

byte bData[] = new byte[132];
nf.opendata("string_data");
nf.getdata(bData);
String string_data = new String(bData);

The aforementioned holds for all strings written as SDS content or as an attribute. SDS or vGroup names do not need this treatment.

Inquiry Routines

Let us compare the C-API and Java-API signatures of the `getinfo()` routine (C) or method (Java):

**C API signature of `getinfo()`**

/* C -API */
NXstatus NXgetinfo(NXhandle handle, int *rank, int iDim[],
int *datatype);

**Java API signature of `getinfo()`**

// Java
void getinfo(int iDim[], int args[]);

The problem is that Java passes arguments only by value, which means they cannot be modified by the method. Only array arguments can be modified. Thus `args` in the `getinfo()` method holds the rank and datatype information passed in separate items in the C-API version. For resolving which one is which, consult a debugger or the API-reference.

The attribute and vGroup search routines have been simplified using Hashtables. The Hashtable returned by `groupdir()` holds the name of the item as a key and the classname or the string SDS as the stored object for the key. Thus the code for a vGroup search looks like this:

**vGroup search**

nf.opengroup(group,nxclass);
h = nf.groupdir();
e = h.keys();
System.out.println("Found in vGroup entry: ");
while(e.hasMoreElements())
{
    vname = (String)e.nextElement();
    vclass = (String)h.get(vname);
    System.out.println(" Item: "+vname+" class: " + vclass);
}

For an attribute search both at global or SDS level the returned Hashtable will hold the name as the key and a little class holding the type and size information as value. Thus an attribute search looks like this in the Java-API:
attribute search

```java
Hashtable h = nf.attrdir();
Enumeration e = h.keys();
while (e.hasMoreElements())
{
    attname = (String) e.nextElement();
    atten = (AttributeEntry) h.get(attname);
    System.out.println("Found global attribute: " + attname +
        " type: "+ atten.type + " ,length: " + atten.length);
}
```

For more information about the usage of the API routines see the reference or the NeXus C-API reference pages. Another good source of information is the source code of the test program which exercises each API routine.

**Known Problems**

These are a couple of known problems which you might run into:

**Memory**

As the Java API for NeXus has to convert between native and Java number types a copy of the data must be made in the process. This means that if you want to read or write 200MB of data your memory requirement will be 400MB! This can be reduced by using multiple `getslab()/putslab()` to perform data transfers in smaller chunks.

**`java.lang.OutOfMemoryException`**

By default the Java runtime has a low default value for the maximum amount of memory it will use. This ceiling can be increased through the `-mxXXm` option to the Java runtime. An example: `java -mx512m ...` starts the Java runtime with a memory ceiling of 512MB.

**Maximum 8192 files open**

The NeXus API for Java has a fixed buffer for file handles which allows only 8192 NeXus files to be open at the same time. If you ever hit this limit, increase the `MAXHANDLE` define in `native/handle.h` and recompile everything.

**On-line Documentation**

The following documentation is browsable online:

1. The API source code
2. A verbose tutorial for the NeXus for Java API.
3. The API Reference.
4. Finally, the source code for the test driver for the API which also serves as a documented usage example.
4.3.5 NAPI IDL Interface

IDL is an interactive data evaluation environment developed by Research Systems - it is an interpreted language for data manipulation and visualization. The NeXus IDL bindings allow access to the NeXus API from within IDL - they are installed when NeXus is compiled from source after being configured with the following options:

```
configure \n   --with-idlroot=/path/to/idl/installation \n   --with-idldlm=/path/to/install/dlm/files/to
```

For further details see the README (https://htmlpreview.github.com/?https://github.com/nexusformat/code/blob/master/bindings/idl/README.html) for the NeXus IDL binding. The source code is stored at https://github.com/nexusformat/code/tree/master/bindings/idl

4.4 Utility API

The NeXus F90 Utility API provides a number of routines that combine the operations of various core API routines in order to simplify the reading and writing of NeXus files. At present, they are only available as a Fortran 90 module but a C version is in preparation.

The utility API comprises the following functional groups:

- Routines to read or write data.
- Routines to find whether or not groups, data, or attributes exist, and to find data with specific signal or axis attributes, i.e. to identify valid data or axes.
- Routines to open other groups to which NXdata items are linked, and to return again.

**line required for use with F90 API**

Any program using the F90 Utility API needs to put the following line near the top of the program:

```
use NXUmodule
```

**Note:** Do not put USE statements for both NXmodule and NXUmodule. The former is included in the latter.
4.4.1 List of F90 Utility Routines

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reading and Writing</strong></td>
<td></td>
</tr>
<tr>
<td>NXUwriteglobals</td>
<td>Writes all the valid global attributes of a file.</td>
</tr>
<tr>
<td>NXUwritegroup</td>
<td>Opens a group (creating it if necessary).</td>
</tr>
<tr>
<td>NXUwritedata</td>
<td>Opens a data item (creating it if necessary) and writes data and its units.</td>
</tr>
<tr>
<td>NXUreaddata</td>
<td>Opens and reads a data item and its units.</td>
</tr>
<tr>
<td>NXUwritehistogram</td>
<td>Opens one dimensional data item (creating it if necessary) and writes</td>
</tr>
<tr>
<td></td>
<td>histogram centers and their units.</td>
</tr>
<tr>
<td>NXUreadhistogram</td>
<td>Opens and reads a one dimensional data item and converts it to histogram</td>
</tr>
<tr>
<td></td>
<td>bin boundaries.</td>
</tr>
<tr>
<td>NXUsetcompress</td>
<td>Defines the compression algorithm and minimum dataset size for subsequent</td>
</tr>
<tr>
<td></td>
<td>write operations.</td>
</tr>
<tr>
<td><strong>Finding Groups, Data, and Attributes</strong></td>
<td></td>
</tr>
<tr>
<td>NXUfindclass</td>
<td>Returns the name of a group of the specified class if it is contained within</td>
</tr>
<tr>
<td></td>
<td>the currently open group.</td>
</tr>
<tr>
<td>NXUfinddata</td>
<td>Checks whether a data item of the specified name is contained within the</td>
</tr>
<tr>
<td></td>
<td>currently open group.</td>
</tr>
<tr>
<td>NXUfindattr</td>
<td>Checks whether the currently open data item has the specified attribute.</td>
</tr>
<tr>
<td>NXUfindsignal</td>
<td>Searches the currently open group for a data item with the specified SIGNAL</td>
</tr>
<tr>
<td></td>
<td>attribute.</td>
</tr>
<tr>
<td>NXUfindaxis</td>
<td>Searches the currently open group for a data item with the specified AXIS</td>
</tr>
<tr>
<td></td>
<td>attribute.</td>
</tr>
<tr>
<td><strong>Finding Linked Groups</strong></td>
<td></td>
</tr>
<tr>
<td>NXUfindlink</td>
<td>Finds another link to the specified NeXus data item and opens the group it</td>
</tr>
<tr>
<td></td>
<td>is in.</td>
</tr>
<tr>
<td>NXUresumelink</td>
<td>Reopens the original group from which NXUfindlink was used.</td>
</tr>
</tbody>
</table>

Currently, the F90 utility API will only write character strings, 4-byte integers and reals, and 8-byte reals. It can read other integer sizes into four-byte integers, but does not differentiate between signed and unsigned integers.

4.5 Building Programs

The install kit provides a utility call nxbuild that can be used to build simple programs:

```
nxbuild -o test test.c
```  

This script links in the various libraries for you and reading its contents would provide the necessary information for creating a separate Makefile. You can also use nxbuild with the example files in the NeXus distribution kit which are installed into `/usr/local/nexus/examples`

Note that the executable name is important in this case as the test program uses it internally to determine the `NXACC_CREATE*` argument to pass to `NXopen`.
building and running a simple NeXus program

```
# builds HDF5 specific test
nxbuild -o napi_test-hdf5 napi_test.c

# runs the test
./napi_test-hdf5
```

NeXus is also set up for pkg-config so the build can be done as:

```
gcc `pkg-config --cflags` `pkg-config --libs` -o test test.c
```

### 4.6 Reporting Bugs in the NeXus API

If you encounter any bugs in the installation or running of the NeXus API, please report them online using our Issue Reporting system. (https://www.nexusformat.org/IssueReporting.html)
NeXus began as a group of scientists with the goal of defining a common data storage format to exchange experimental results and to exchange ideas about how to analyze them.

The NeXus Scientific Community provides the scientific data, advice, and continued involvement with the NeXus standard. NeXus provides a forum for the scientific community to exchange ideas in data storage.

The NeXus International Advisory Committee (NIAC) supervises the development and maintenance of the NeXus common data format for neutron, X-ray, and muon science through the NeXus class definitions and oversees the maintenance of the NeXus Application Programmer Interface (NAPI) as well as the technical infrastructure.

There are several mechanisms in place in order to coordinate the development of NeXus with the larger community.

### 5.1 NeXus Webpage

First of all, there is the NeXus webpage, [https://www.nexusformat.org/](https://www.nexusformat.org/), which provides all kinds of information, including membership, minutes, and discussions from the meetings of the NIAC, Code Camps, and Tele Conferences, as well as some proposed designs for consideration by NeXus.

The webpage is kept with a number of other repositories in the nexusformat.org Github organisation [https://github.com/nexusformat/](https://github.com/nexusformat/). As for all of these repositories, pull requests to correct or improve the content or code are always welcome!

### 5.2 Contributed Definitions

The community is encouraged to provide new definitions ([Base Class Definitions](https://www.nexusformat.org/niac.html) or [Application Definitions](https://www.nexusformat.org/niac.html)) for consideration in the NeXus standard. These community contributions will be entered in the [Contributed Definitions](https://www.nexusformat.org/niac.html) and will be curated according to procedures set forth by the [NIAC: The NeXus International Advisory Committee](https://www.nexusformat.org/niac.html).

### 5.3 Other Ways NeXus Coordinates with the Scientific Community

#### 5.3.1 NIAC: The NeXus International Advisory Committee

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.

---

\(^1\) For more details about the NIAC constitution, procedures, and meetings, refer to the NIAC web page: [https://www.nexusformat.org/NIAC.html](https://www.nexusformat.org/NIAC.html)

The members of the NIAC may be reached by email: nexus-committee@nexusformat.org
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 (fields) 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 the maintenance of the NeXus Application Programming Interface 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 reach compatibility.

The committee will meet at least once every other 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 community. This is intended to be a working meeting with a small group.

5.3.2 NeXus Mailing List

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

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

Note: Subscription to nexus@nexusformat.org does not subscribe you automatically to any other NeXus mailing list.

5.3.3 NeXus International Advisory Committee (NIAC) Mailing List

This list contains discussions of the NIAC: The NeXus International Advisory Committee, 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 limited to the committee members, but the archives are public. The NIAC mailing list is for communications specific to NIAC and not for public contribution. General discussions should be held in the public mailing list.

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

Note: Subscription to nexus-committee@nexusformat.org does not subscribe you automatically to any other NeXus mailing list.
5.3.4 **NeXus Video Conference Announcements**

There are video conferences on NeXus roughly twice a month. Agenda and joining details are posted on the webpage: [https://www.nexusformat.org/Teleconferences.html](https://www.nexusformat.org/Teleconferences.html) In addition calendar invites are sent to this list. NeXus-Tech used to be used for discussions in the past. Now the list is moderated to only allow communication related to holding meetings. All other traffic should go to the main list nexus@nexusformat.org

- **List Address:** nexus-tech@nexusformat.org
- **Subscriptions:** [http://lists.nexusformat.org/mailman/listinfo/nexus-tech](http://lists.nexusformat.org/mailman/listinfo/nexus-tech)

5.3.5 **NeXus Developers Mailing List (retired)**

This mailing list was for discussions concerning the technical development of NeXus (the Definitions, NXDL, and the NeXus Application Program Interface). There was, however, much overlap with the general NeXus mailing list and so this separate list was closed in October 2012, but the archive of previous posting is still available.

- **Archive:** [http://lists.nexusformat.org/pipermail/nexus-developers](http://lists.nexusformat.org/pipermail/nexus-developers)

5.3.6 **NeXus Repositories**

NeXus NXDL class definitions (both base classes and application definitions) and the NeXus code library source are held in a pair of git repositories on GitHub.

The repositories are world readable. You can browse them directly:

- **NeXus code library and applications**
  - [https://github.com/nexusformat/code](https://github.com/nexusformat/code)
- **NeXus NXDL class definitions**
  - [https://github.com/nexusformat/definitions](https://github.com/nexusformat/definitions)
- **NeXus GitHub organization**
  - [https://github.com/nexusformat](https://github.com/nexusformat)

If you would like to contribute (thank you!), the normal GitHub procedure of forking the repository and generating pull requests should be used.

Please report any problems via the Issue Reporting system.

5.3.7 **NeXus Issue Reporting**

NeXus is using GitHub ([https://github.com](https://github.com)) as source code repository and for problem reporting. The issue reports (see View current issues below) are used to guide the NeXus developers in resolving problems as well as implementing new features.
NeXus Code (NAPI, Library, and Applications)

Report a new issue
https://github.com/nexusformat/code/issues/new

View current issues
https://github.com/nexusformat/code/issues

Timeline (recent ticket and code changes)
https://github.com/nexusformat/code/pulse

NeXus Definitions (NXDL base classes and application definitions)

Report a new issue
https://github.com/nexusformat/definitions/issues/new

View current issues
https://github.com/nexusformat/definitions/issues

Timeline (recent ticket and definition changes)
https://github.com/nexusformat/definitions/pulse
This section describes how to install the NeXus API and details the requirements. The NeXus API is distributed under the terms of the GNU Lesser General Public License version 3.

The source distribution of NAPI can be downloaded from the release page of the associated GitHub project. Instructions how to build the code can be found in the INSTALL.rst file shipped with the source distribution. In case you need help, feel free to contact the NeXus mailing list: http://lists.nexusformat.org/mailman/listinfo/nexus

6.1 Precompiled Binary Installation

6.1.1 Linux RPM Distribution Kits

An installation kit (source or binary) can be downloaded from: https://github.com/nexusformat/code/releases/tag/4.3.0

A NeXus binary RPM (nexus-*.i386.rpm) contains ready compiled NeXus libraries whereas a source RPM (nexus-*.src.rpm) needs to be compiled into a binary RPM before it can be installed. In general, a binary RPM is installed using the command

```
rpm -Uvh file.i386.rpm
```

or, to change installation location from the default (e.g. /usr/local) area, using

```
rpm -Uvh --prefix /alternative/directory file.i386.rpm
```

If the binary RPMs are not the correct architecture for you (e.g. you need x86_64 rather than i386) or the binary RPM requires libraries (e.g. HDF4) that you do not have, you can instead rebuild a source RPM (.src.rpm) to generate the correct binary RPM for you machine. Download the source RPM file and then run

```
rpmbuild --rebuild file.src.rpm
```

This should generate a binary RPM file which you can install as above. Be careful if you think about specifying an alternative buildroot for rpmbuild by using --buildroot option as the “buildroot” directory tree will get remove (so --buildroot / is a really bad idea). Only change buildroot it if the default area turns out not to be big enough to compile the package.

If you are using Fedora, then you can install all the dependencies by typing

```
yum install hdf hdf-devel hdf5 hdf5-devel mxml mxml-devel
```
6.1.2 Microsoft Windows Installation Kit

A Windows MSI based installation kit is available and can be downloaded from: https://github.com/nexusformat/code/releases/tag/4.3.0

6.1.3 Mac OS X Installation Kit

An installation disk image (.dmg) can be downloaded from: https://github.com/nexusformat/code/releases/tag/4.3.0

6.2 Source Installation

6.2.1 NeXus Source Code Distribution

The source code distribution can be obtained from GitHub. One can either checkout the git repositories to get access to the most recent development code. To clone the definitions repository use

```
$ git clone https://github.com/nexusformat/definitions.git definitions
```

or for the NAPI

```
$ git clone https://github.com/nexusformat/code.git code
```

For release tarballs go to the release page for the NAPI or the definitions. For the definitions it is currently recommended to work directly with the Git repository as the actual release is rather outdated.

Instructions how to build the NAPI code can be found either on the GitHub project website or in the `README.rst` file shipped with the source distribution.

6.3 Releases

The NeXus definitions are expected to evolve. The evolution is marked as a series of releases which are snapshots of the repository (and current state of the NeXus standard). Each new release of the definitions will be posted to the definitions GitHub repository and announced to the community via the NeXus mailing list: nexus@nexusformat.org

6.3.1 NeXus definitions

Releases of the NeXus definitions are listed on the GitHub web site: https://github.com/nexusformat/definitions/releases

Release Notes

Detailed notes about each release (start with v3.3) are posted to the definitions GitHub wiki: https://github.com/nexusformat/definitions/wiki/Release-Notes
Release Process

The process to make a new release of the NeXus definitions repository is documented in the repository’s GitHub wiki: https://github.com/nexusformat/definitions/wiki/Release-Procedure.

The release process starts by creating a GitHub [Milestone](https://help.github.com/articles/tracking-the-progress-of-your-work-with-milestones/) for the new release. Milestones for the NeXus definitions repository are available on the GitHub site: https://github.com/nexusformat/definitions/milestones

Versioning (Tags)

Versioning of each of the NXDL files, as well as the complete set of NXDL files is now described in the wiki\(^1\) of the NeXus definitions repository\(^2\). Please see that wiki for complete information.

In case you need help, feel free to contact the NeXus Mailing List:

- **Archives**
  http://lists.nexusformat.org/mailman/listinfo/nexus

- **email**
  nexus@nexusformat.org

---

\(^1\) Release Procedure: https://github.com/nexusformat/definitions/wiki/Release-Procedure

\(^2\) Definitions repository: https://github.com/nexusformat/definitions

6.3. Releases
NEXUS UTILITIES

There are many utilities available to read, browse, write, and use NeXus data files. Some are provided by the NeXus technical group while others are provided by the community. Still, other tools listed here can read or write one of the low-level file formats used by NeXus (HDF5, HDF4, or XML).

Furthermore, there are specific examples of code that can read, write, (or both) NeXus data files, given in the section *Language APIs for NeXus and HDF5*.

The NIAC welcomes your continued contributions to this documentation.

Please note that NeXus maintains a repository of example data files which you may browse and download. There is a cursory analysis of every file in this repository as to whether it can be read as HDF5 or NeXus HDF5. The analysis code, which serves as yet another example reader, is made using python and h5py.

7.1 Utilities supplied with NeXus

Most of these utility programs are run from the command line. It will be noted if a program provides a graphical user interface (GUI). Short descriptions are provided here with links to further information, as available.

**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* attribute will be printed when showing the value of the specified

---

1 https://github.com/nexusformat/exampledata
2 https://github.com/nexusformat/exampledata/blob/master/critique.md
3 https://github.com/nexusformat/exampledata/blob/master/critique.py
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.

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.)

7.2 Validation

The list of applications below are for validating NeXus files. The list is not intended to be a complete list of all available packages.

cnxvalidate

NeXus validation tool written in C (not via NAPI).

Its dependencies are libxml2 and the HDF5 libraries, version 1.8.9 or better. Its purpose is to validate HDF5 files against NeXus application definitions.

See the program documentation for more details: https://github.com/nexusformat/cnxvalidate.git

punx

Python Utilities for NeXus HDF5 files

punx can validate both NXDL files and NeXus HDF5 data files, as well as print the structure of any HDF5 file, even non-NeXus files.

NOTE: project is under initial construction, not yet released for public use, but is useful in its present form (version 0.2.5).

punx can show the tree structure of any HDF5 file. The output is more concise than that from h5dump.

See the program documentation for more details: https://punx.readthedocs.io

7.3 Other Utilities

NeXus Constructor (https://github.com/ess-dmsc/nexus-constructor)

The NeXus Constructor facilitates constructing NeXus files in which to record data from experiments at neutron science facilities. This includes all supporting metadata typically required to perform analysis of such experiments, including instrument geometry information.

nxdl_to_hdf5.py (https://github.com/nexusformat/exampledata/tree/master/nxdl)
nxdl_to_hdf5.py is a Python script that reads the NeXus definition files (files ending with .nxdl.xml) and creates example Python scripts as well as HDF5 files for each definition. There are generated example scripts of each application definition for both h5py and nexusformat. Currently, only application definitions and some contributed_definitions are supported as the code depends on the existence of an NXentry in the definition.
7.4 Data Analysis

The list of applications below are some of the utilities that have been developed (or modified) to read/write NeXus files as a data format. It is not intended to be a complete list of all available packages.

DAVE is an integrated environment for the reduction, visualization and analysis of inelastic neutron scattering data. It is built using IDL (Interactive Data Language) from ITT Visual Information Solutions.

**DAWN** ([http://www.dawnsci.org](http://www.dawnsci.org))
The Data Analysis WorkbeNch (DAWN) project is an eclipse based workbench for doing scientific data analysis. It offers generic visualisation, and domain specific processing.

**GDA** ([http://www.opengda.org](http://www.opengda.org))
The GDA project is an open-source framework for creating customised data acquisition software for science facilities such as neutron and X-ray sources. It has elements of the DAWN analysis workbench built in.

Gumtree is an open source project, providing a graphical user interface for instrument status and control, data acquisition and data reduction.

**IDL** ([https://www.harrisgeospatial.com/docs/using_idl_home.html](https://www.harrisgeospatial.com/docs/using_idl_home.html))
IDL is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation.

**IGOR Pro** ([http://www.wavemetrics.com/](http://www.wavemetrics.com/))
IGOR Pro is an extraordinarily powerful and extensible scientific graphing, data analysis, image processing and programming software tool for scientists and engineers.

The Integrated Spectral Analysis Workbench software project (ISAW) is a Platform-Independent system Data Reduction/Visualization. ISAW can be used to read, manipulate, view, and save neutron scattering data. It reads data from IPNS run files or NeXus files and can merge and sort data from separate measurements.

**LAMP** ([http://www.ill.eu/data_treat/lamp/](http://www.ill.eu/data_treat/lamp/))
LAMP (Large Array Manipulation Program) is designed for the treatment of data obtained from neutron scattering experiments at the Institut Laue-Langevin. However, LAMP is now a more general purpose application which can be seen as a GUI-laboratory for data analysis based on the IDL language.

**Mantid** ([http://www.mantidproject.org/](http://www.mantidproject.org/))
The Mantid project provides a platform that supports high-performance computing on neutron and muon data. It is being developed as a collaboration between Rutherford Appleton Laboratory and Oak Ridge National Laboratory.

**MATLAB** ([http://www.mathworks.com/](http://www.mathworks.com/))
MATLAB is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation.

**NeXpy** ([http://nexpy.github.io/nexpy/](http://nexpy.github.io/nexpy/))
The goal of NeXpy is to provide a simple graphical environment, coupled with Python scripting capabilities, for the analysis of X-Ray and neutron scattering data. (It was decided at the NIAC 2010 meeting that a large portion of this code would be adopted in the future by NeXus and be part of the distribution)

The silx project aims to provide a collection of Python packages to support the development of data assessment, reduction and analysis at synchrotron radiation facilities. In particular it provides tools to read, write and visualize NeXus HDF5 files.
OpenGENIE (http://www.opengenie.org/)
A general purpose data analysis and visualisation package primarily developed at the ISIS Facility, Rutherford Appleton Laboratory.

PyMCA (http://pymca.sourceforge.net/)
PyMca is a ready-to-use, and in many aspects state-of-the-art, set of applications implementing most of the needs of X-ray fluorescence data analysis. It also provides a Python toolkit for visualization and analysis of energy-dispersive X-ray fluorescence data. Reads, browses, and plots data from NeXus HDF5 files.

spec2nexus (https://spec2nexus.readthedocs.io)
(Python code) Converts SPEC data files and scans into NeXus HDF5 files. (Note the h5toText tool mentioned here previously is no longer available from the spec2nexus project. The code has been moved into the punx project: https://punx.readthedocs.io/.)

spec2nexus provides libraries:
- spec2nexus.spec: python binding to read SPEC data files
- spec2nexus.eznx: (Easy NeXus) supports writing NeXus HDF5 files using h5py

7.5 HDF Tools

Here are some of the generic tools that are available to work with HDF files. In addition to the software listed here there are also APIs for many programming languages that will allow low level programmatic access to the data structures.

h5wasm (https://github.com/usnistgov/h5wasm):
A WebAssembly port of the HDF5 C library, which allows reading and writing HDF5 files from JavaScript (i.e. no need for a back-end server at all).

H5Web (https://github.com/silx-kit/h5web):
H5Web is a toolkit for exploring and visualising HDF5 files and, more generally, for visualizing data. It is based on React, and WebGL. These projects make use of H5Web:
- jupyterlab-h5web: extension for JupyterLab
- vscode-h5web: extension for Microsoft Visual Studio Code Editor
- On-line visualization with NeXus file (using h5wasm): simple_example_basic.nexus.hdf5
- H5Web demonstration site

HDF Group tools (https://portal.hdfgroup.org/display/support/Downloads)
Various tools are available from the HDF Group. These are usually shipped with the HDF5 kits but are also available for download separately. The HDF5 source code (https://github.com/HDFGroup/hdf5) is available on GitHub.

HDFexplorer (http://www.space-research.org/)
A data visualization program that reads Hierarchical Data Format files (HDF, HDF-EOS and HDF5) and also netCDF data files.

HDFview (http://www.hdfgroup.org)
A Java based GUI for browsing (and some basic plotting) of HDF files.

tiled (https://blueskyproject.io/tiled/)
A data access service for data-aware portals and data science tools, provides a way to browse and visualize HDF5 files.

---

4 SPEC: http://www.certif.com
7.6 Language APIs for NeXus and HDF5

Collected here are some of the tools identified\(^5\) as a result of a simple question asked at the 2018 Nobugs conference: *Are there examples of code that reads NeXus data?* Some of these are very specific to an instrument or application definition while others are more generic. The lists below are organized by programming language, yet some collections span several languages so they are listed in the section *Language API: mixed.*

Note these example listed in addition to the many examples described here in the manual, in section :Examples.

7.6.1 Language API: F77

- **POLDI**: [poldi.zip\(^6\)](https://github.com/nexusformat/definitions/files/4107360/poldi.zip) contains: A F77 reading routine using NAPI for POLDI at SINQ PSI - an example of a file which it reads

7.6.2 Language API: IDL

- **aXis2000\(^7\)**, with the NeXus-specific IDL code in the [read_nexus.pro\(^8\)](http://unicorn.chemistry.mcmaster.ca/axis/aXis2000.zip), reads *NXstxm*

7.6.3 Language API: IgorPro

- **HDF5gateway\(^9\)** makes it easy to read a HDF5 file (including NeXus) into an IgorPro\(^\text{10}\) folder, including group and dataset attributes, such as a NeXus data file, modify it, and then write the folder structure back out.

7.6.4 Language API: Java

- **Dawn\(^\text{11}\)** has java code to read\(^\text{12}\) and write\(^\text{13}\) HDF5 NeXus files (generic NeXus, not tied to specific application definitions).

- **NXreader.zip\(^\text{14}\)** is java code which reads NeXus files into *ImageJ*. It uses the Java-hdf interface to HDF5. It tries to do a good job locating the image dataset by NeXus conventions. But it uses the old style conventions.

---

\(^{5}\) https://github.com/nexusformat/definitions/issues/630  
\(^{6}\) https://github.com/nexusformat/definitions/files/4107360/poldi.zip  
\(^{7}\) http://unicorn.chemistry.mcmaster.ca/aXis2000.html  
\(^{8}\) read_nexus.pro: http://unicorn.chemistry.mcmaster.ca/axis/aXis2000.zip  
\(^{9}\) https://github.com/prjemian/hdf5gateway  
\(^{10}\) IgorPro: https://wavemetrics.com  
\(^{11}\) https://dawnsri.org/  
\(^{14}\) https://github.com/nexusformat/definitions/files/4107439/NXreader.zip
7.6.5 Language API: Python

- **Dials**\(^\text{15}\) has python (and some C++) code for reading \textit{NXmx}\(^\text{16}\)
  - \textit{cctbx.xfel} code for writing \textit{NXmx} master files for JF16M at SwissFEL
- **h5py**\(^\text{Page 624, 18}\)
  HDF5 for Python (h5py) is a general-purpose Python interface to HDF5.
- **Mantis**\(^\text{19}\), with NeXus-specific python code\(^\text{20}\), reads \textit{NXstxm}
- **nexusformat**\(^\text{21}\) NeXus package for Python
  Provides an API to open, create, plot, and manipulate NeXus data.
- **SasView**\(^\text{22}\) has python code to read\(^\text{23}\) and write\(^\text{24}\) \textit{NXcanSAS}

7.6.6 Language API: mixed

- **FOCUS**: \texttt{focus.zip}\(^\text{25}\) contains:
  - An example FOCUS file
  - \texttt{focusreport}: A h5py program which skips through a list of files and prints statistics
  - \texttt{focusreport.tcl}, same as above but in Tcl using the Swig generated binding to NAPI
  - \texttt{i80.f} contains a F77 routine for reading FOCUS files into Ida. The routine is \texttt{RRT_in_Foc}.
- **ZEBRA**: \texttt{zebra.zip}\(^\text{26}\) contains:
  - an example file
  - \texttt{zebra-to-ascii}, a h5py script which dumps a zebra file to ASCII
  - \texttt{TRICSReader.*} for reading ZEBRA files in C++ using C-NAPI calls

---

\(^\text{15}\) https://dials.github.io/
\(^\text{16}\) read: https://github.com/cctbx/dxtbx/blob/master/format/nexus.py
\(^\text{17}\) write: https://github.com/cctbx/cctbx_project/blob/master/xfel/jf16m_cxigeom2nexus.py
\(^\text{18}\) http://docs.h5py.org
\(^\text{19}\) Mantis: http://spectromicroscopy.com/
\(^\text{20}\) python code: https://bitbucket.org/mlerotic/spectromicroscopy/src/default/
\(^\text{21}\) https://github.com/nexpy/nexusformat
\(^\text{22}\) https://www.sasview.org/
\(^\text{23}\) read: https://github.com/SasView/sasview/blob/master/src/sas/sascalc/dataloader/readers/cansas_reader_HDF5.py
\(^\text{24}\) write: https://github.com/SasView/sasview/blob/master/src/sas/sascalc/file_converter/nxcansas_writer.py
\(^\text{25}\) https://github.com/nexusformat/definitions/files/4107386/focus.zip
\(^\text{26}\) https://github.com/nexusformat/definitions/files/4107416/zebra.zip
BRIEF HISTORY OF NEXUS

Two things to note about the development and history of NeXus:

- All efforts on NeXus have been voluntary except for one year when we had one full-time worker.
- The NIAC has already discussed many matters related to the format.

2022-07:
- release v2022.06 <https://github.com/nexusformat/definitions/wiki/releasenotes__v2022.06> of NeXus Definitions

2020-10:

2020-01:

2018-05:
- release v2018.5 <https://github.com/nexusformat/definitions/wiki/releasenotes__v2018.5> of NeXus Definitions
- #597 changed versioning scheme and procedures

2017-07:
- release 3.3 <https://github.com/nexusformat/definitions/wiki/releasenotes__v3.3> of NeXus Definitions

2016-10:
- release 3.2 <https://github.com/nexusformat/definitions/releases/tag/v3.2> of NeXus Definitions

2014-12:
- The NIAC approves a new method to identify the default data to be plotted, applying attributes at the group level to the root of the HDF5 tree, and the NXentry and NXdata groups. See the description in Associating plottable data using attributes applied to the NXdata group and the proposal: https://www.nexusformat.org/2014_How_to_find_default_data.html

2012-05:
- first release (3.1.0) of NXDL (NeXus Definition Language)

2010-01:
- NXDL presented to ESRF HDF5 workshop on hyperspectral data

2009-09:
- NXDL and draft NXsas (base class) presented to canSAS at SAS2009 conference
2009-04:
NeXus API version 4.2.0 is released with additional C++, IDL, and python/numpy interfaces.

2008-10:
NXDL: The NeXus Definition Language is defined. Until now, NeXus used another XML format, meta-DTD, for defining base classes and application definitions. There were several problems with meta-DTD, the biggest one being that it was not easy to validate against it. NXDL was designed to circumvent these problems. All current base classes and application definitions were ported into the NXDL.

2007-10:
NeXus API version 4.1.0 is released with many bug-fixes.

2007-05:
NeXus API version 4.0.0 is released with broader support for scripting languages and the feature to link with external files.

2005-07:
The community asked the NeXus team to provide an ASCII based physical file format which allows them to edit their scientific results in emacs. This lead to the development of a XML NeXus physical format. This was released with NeXus API version 3.0.0.

2003-10:
In 2003, NeXus had arrived at a stage where informal gatherings of a group of people were no longer good enough to oversee the development of NeXus. This lead to the formation of the NeXus International Advisory Committee (NIAC) which strives to include representatives of all major stake holders in NeXus. A first meeting was held at CalTech. Since 2003, the NIAC meets every year to discuss all matters NeXus.

2003-06:
Przemek Klosowski, Ray Osborn, and Richard Riedel received the only known grant explicitly for working on NeXus from the Systems Integration for Manufacturing Applications (SIMA) program of the National Institute of Standards and Technology (NIST). The grant funded a person for one year to work on community wide infrastructure in NeXus.

2002-09:
NeXus API version 2.0.0 is released. This version brought support for the new version of HDF, HDF5, released by the HDF group. HDF4 imposed limits on file sizes and the number of objects in a file. These issues were resolved with HDF5. The NeXus API abstracted the difference between the two physical file formats away form the user.

2001-summer:
MLNSC at LANL started writing NeXus files to store raw data

1997-07:
SINQ at PSI started writing NeXus files to store raw data.

1996-10:
At SoftNeSS 1996 (at ANL), after reviewing the different scientific data formats discussed, it was decided to use HDF4 as it provided the best grouping support. The basic structure of a NeXus file was agreed upon. the various data format proposals were combined into a single document by Przemek Klosowski (NIST), Mark Könnecke (then ISIS), Jonathan Tischler (ORNL and APS/ANL), and Ray Osborn (IPNS/ANL) coauthored the first proposal for the NeXus scientific data standard.¹

1996-08:
The HDF-4 API is quite complex. Thus a NeXus Abstract Programmer Interface NAPI was released which simplified reading and writing NeXus files.

1995-09:
At SoftNeSS 1995 (at NIST), three individual data format proposals by Przemek Klosowski (NIST), Mark Kön-

¹ https://www.nexusformat.org/pdfs/NeXus_Proposal.pdf
necke (then ISIS), and Jonathan Tischler (ORNL and APS/ANL) were joined to form the basis of the current NeXus format. At this workshop, the name NeXus was chosen.

1994-10:
Ray Osborn convened a series of three workshops called SoftNeSS. In the first meeting, Mark Könnecke and Jon Tischler were invited to meet with representatives from all the major U.S. neutron scattering laboratories at Argonne National Laboratory to discuss future software development for the analysis and visualization of neutron data. One of the main recommendations of SoftNeSS’94 was that a common data format should be developed.

1994-08:
Jonathan Tischler (ORNL) proposed an HDF-based format\(^2\) as a standard for data storage at APS

1994-06:
Mark Könnecke (then ISIS, now PSI) made a proposal using netCDF\(^3\) for the European neutron scattering community while working at ISIS

\(^2\) https://www.nexusformat.org/pdfs/Proposed_Data_Standard_for_the_APS.pdf
\(^3\) https://www.nexusformat.org/pdfs/European-Formats.pdf
CHAPTER
NINE

ABOUT THESE DOCS

9.1 Authors

Pete R. Jemian, Documentation Editor:
<jemian@anl.gov>, Advanced Photon Source, Argonne National Laboratory, Argonne, IL, USA,

Frederick Akeroyd:
<freddie.akeroyd@stfc.ac.uk>, Rutherford Appleton Laboratory, Didcot, UK,

Stuart I. Campbell:
<campbellsi@ornl.gov>, Oak Ridge National Laboratory, Oak Ridge, TN, USA,

Przemek Klosowski:
<przemek.klosowski@nist.gov>, U. of Maryland and NIST, Gaithersburg, MD, USA,

Mark Könnecke:
<Mark.Koenecke@psi.ch>, Paul Scherrer Institut, CH-5232 Villigen PSI, Switzerland,

Ray Osborn:
<rosborn@anl.gov>, Argonne National Laboratory, Argonne, IL, USA,

Peter F. Peterson:
<petersonpf@ornl.gov>, Spallation Neutron Source, Oak Ridge, TN, USA,

Tobias Richter:
<Tobias.Richter@esss.se>, European Spallation Source, Lund, Sweden,

Joachim Wuttke:
<j.wuttke@fz-juelich.de>, Forschungszentrum Jülich, Jülich Centre for Neutron Science at Heinz Maier-Leibnitz Zentrum Garching, Germany.

9.2 Colophon

These docs (manual and reference) were produced using Sphinx (http://sphinx-doc.org). The source for the manual shows many examples of the structures used to create the manual. If you have any questions about how to contribute to this manual, please contact the NeXus Documentation Editor (Pete Jemian <jemian@anl.gov>).

Note: The indentation is very important to the syntax of the restructured text manual source. Be careful not to mix tabs and spaces in the indentation or the manual may not build properly.
9.3 Revision History

Browse the most recent Issues on the GitHub repository: https://github.com/nexusformat/definitions/pulse/monthly

9.4 Copyright and Licenses

Published by NeXus International Advisory Committee, https://www.nexusformat.org
Copyright (C) 1996-2022 NeXus International Advisory Committee (NIAC)
The NeXus manual is licensed under the terms of the GNU Free Documentation License version 1.3.

- **download FDL**
  - **GNU**
    - http://www.gnu.org/licenses/fdl-1.3.txt

The code examples in the NeXus manual are licensed under the terms of the GNU Lesser General Public License version 3.

- **download LGPL**
  - **GNU**
    - http://www.gnu.org/licenses/lgpl-3.0.txt

---

**Publishing Information**

This manual built Sep 21, 2022.

See also:

This document is available in these formats online:

- **HTML**
  - https://manual.nexusformat.org/

- **PDF**

A very brief overview (title: *NeXus for the Impatient*) is also available (separate from the manual).

- **HTML**
  - https://manual.nexusformat.org/impatient/

- **PDF**
aberration (base class), see NXaberration, 435
absorbing_material (field), 190, 231
absorption_cross_section (field), 176
ac_line_sync (field), 555
acceleration_time (field), 273
accepted_photon_beam_divergence (field), 184
accepting_aperture (field), 320
acp_aperture_em (base class), see NXaperture_em, 436
API, see NAPI
  F77; POLDI, 623
  IDL; aXis2000, 623
  IgorPro; HDF5gateway, 623
  java; Dawn, 623
  java; NXreader.zip, 623
  mixed; FOCUS, 624
  mixed; ZEBRA, 624
  Python; Dials, 624
  Python; h5py, 624
  Python; Mantis, 624
  Python; nexusformat, 624
  Python; SasView, 624
apm (application definition), see NXapm, 437
application definition, 321
applied (field), 443, 457, 465, 473, 540, 554
archive (application definition), see NXarchive, 323
arpes (application definition), see NXarpes, 327
arrival_time_pairs (field), 446
atom_types (field), 442, 489, 502
attached_to (field), 298
attenuator (base class), see NXattenuator, 175
attenuator_transmission (field), 176, 363, 431
attribute, see field attribute, see group attribute, see file attribute, 139, see NXDL attribute
  HDF, 57
  NXclass, 40
attribute (NXDL element), 142
attributeType (NXDL data type), 150
author (field), 264, 559, 568
authors, 629
automatic plotting, see plotting
auxiliary_signals (file attribute), 199
average_value (field), 252, 558, 559, 567
average_value_error (field), 252, 558, 559, 567
average_value_errors (field), 252, 558, 559, 567, 568
axes (attribute), 55, 199
axes (field attribute), 201, 468
axes (file attribute), 200
axes (group attribute), 212, 245, 468, 469, 492
axis, 56
axis (field attribute), 201, 204, 205
AXISNAME (field), 201, 313
AXISNAME_end (field), 314
AXISNAME_increment_set (field), 314
AXISNAME_indices (file attribute), 200
azimuthal (field), 389
azimuthal_angle (field), 194, 206, 282, 357, 381, 385, 401, 404, 406, 560, 569
azimuthal_width (field), 389
background_mean (field), 281
band_contrast (field), 522
band_slope (field), 522
bands (field), 522
bandwidth (field), 247
base_class, 171
base_temperature (field), 444
baseline_reference (field attribute), 591–593
basicComponent (NXDL data type), 163
beam (base_class), see NXbeam, 177
beam_center Derived (field), 365
beam_center_x (field), 207, 342, 365, 381, 385, 431, 469, 594
beam_center_y (field), 207, 342, 365, 382, 385, 431, 469, 594
beam_position (field), 221
beam_shape (field), 344
beam_size_x (field), 344
beam_size_y (field), 344
beam_stop (base_class), see NXbeam_stop, 182
beamline_distance (field), 479, 533, 549, 556, 576, 582
bend_angle_x (field), 244, 254
bend_angle_y (field), 244, 254
bending_magnet (base_class), see Nxbending_magnet, 184
bending_radius (field), 184
bias_voltage (field), 527, 587
bibtext (field), 188
bin_ends (field), 451
binary data, 47, see NX_BINARY
binary executable, see NAPI installation
binning (field), 524
bit_depth_readout (field), 210, 367
blade_spacing (field), 190
blade_thickness (field), 190
block (field), 552
bounding_box (field), 281
bragg_angle (field), 194
brightness (field), 514
broadening (field), 491
browser, 16, 619
bunch_distance (field), 306
bunch_length (field), 306

c

code examples, 81
c_1_0 (field), 435
c_1_2_a (field), 435
c_1_2_b (field), 435
c_2_1_a (field), 435

c_2_1_b (field), 435
c_2_3_a (field), 435
c_2_3_b (field), 435
c_3_0 (field), 435
c_3_2_a (field), 435
c_3_2_b (field), 436
c_3_4_a (field), 436
c_3_4_b (field), 436

c_5_0 (field), 436
calibrated_axis (field), 457, 540
calibrated_tof (field), 448
calibration (base_class), see NXcalibration, 457
calibration_angle_of_incidence (field), 485
calibration_data (field), 485
calibration_data_type (field), 485
calibration_date (field), 207
calibration_sample (field), 484
calibration_status (field), 484
calibration_style (field), 555
calibration_time (field), 484
calibration_wavelength (field), 485
camera_length (field), 543
canSAS, 330
canSAS (application definition), see NXcanSAS, 330
canSAS_class (group attribute), 332, 333, 340–342, 344–346
capability (field), 536
capillary (base_class), see NXcapillary, 186
category (NXDL attribute), 68
cdeform_field (field), 474
center (field), 555
center_energy (field), 508
central_stop_diameter (field), 238
central_stop_material (field), 238
central_stop_thickness (field), 238
chamber (base_class), see NXchamber, 458
changer_position (field), 289, 562, 571
charge_state (field), 452, 530
check_sum (field attribute), 204
chemical_formula (field), 192, 233, 288, 295, 326, 462, 540
chi (field), 424
choice, 140
choice (NXDL element), 142
choiceType (NXDL data type), 160
CIF, 28
cite (base_class), see NXCITE, 187
class definitions, 169, see base_class, see application
definition, see contributed definition
class path, 21
cnxvalidate (utility), 620
coating_material (field), 234, 242, 244, 254
coating_roughness (field), 235, 242, 244, 254
coefficients (field), 457
collection (base class), see NXcollection, 188
collection_description (field), 223, 309, 324
collection_identifier (field), 223, 309, 324, 558, 567
collection_time (field), 224, 310, 324
collection_title (field), 558, 567
collectioncolumn (base class), see NXcollectioncolumn, 489
collimator (base class), see NXcollimator, 189
column_names (field), 490
column1 (field), 559, 568
comment (field), 225, 310
community, 611
comp_current (field), 237
comp_turns (field), 237
cOMP_company (field), 483
component (field), 290
component_index (field), 240
composition (field), 272
concentration (field), 290
confidence_index (field), 523
configuration (field), 225, 310
constitution, 77, 611
construction_year (field), 483
container (base class), see NXcontainer, 460
contribute, 77
contribution_definition, 432, 611
corrector_record (field), 273
conversion, 619
coordinate systems, 31
CIF, 31
IUCr, 26
McStas, 26, 31
NeXus, 25
NeXus polar coordinate, 31
spherical polar, 32
transformations, 26
coordinate_system_set (base class), see NXcoordinate_system_set, 463
copyright, 630
corrector_cs (base class), see NXcorrector_cs, 464
count (field), 552
count_time (field), 207, 260, 365, 594
countrate_correction_applied (field), 209, 367
countrate_correction_lookup_table (field), 209
counts (field), 450, 451, 579, 581
coupled (field), 257
coupling_material (field), 257, 561, 570
crate (field), 206
creator (field), 286
creator_version (field), 286
critical_energy (field), 184
cryocoolant (field), 534
cryostat_temperature (field), 534
crystal (base class), see NXcrystal, 191
csg (base class), see NXcsg, 466
cue_index (field), 230, 253
cue_timestamp_zero (field), 230, 252
current (field), 305, 472, 514, 532
curvature (field), 320
curvature_horizontal (field), 194
curvature_vertical (field), 194
cut_angle (field), 193
cxi_ptycho (application definition), see NXcxi_ptycho, 467
cylinders (field), 197
cylindrical (field), 320
cylindrical_geometry (base class), see NXcylindricalGeometry, 197
cylindrical_orientation_angle (field), 194
D
d (field), 278
d_spacing (field), 193
data
multi-dimensional, 52
data (base class), see NXdata, 198
DATA (field), 201
data analysis software, 620
data field, see field
data group, see group
data item, see field
data object, see field
data set, see field
data type, 165
data_correction (field), 483
data_error (field), 490
data_errors (field), 204
data_identifier (field), 489
data_origin (field), 217, 369
data_size (field), 217, 369
data_stride (field), 369
data_type (field), 489
data_x_time_of_flight (field), 569
data_x_y (field), 560, 569
data_y_time_of_flight (field), 569
dataset, see field
date (field), 264, 275, 345, 414, 419, 559, 568
date and time, 47
DAVE (data analysis software), 621
DAWN (data analysis software), 621
dead_time (field), 206, 365
default, 49
default (group attribute), 332
default attribute value, 49, 223, 286, 309
default plot, see plotting
definition (NXDL data type), 151
definition (NXDL element), 68, 142
definition_local (field), 224, 309
definitionType (NXDL data type), 151
definitionTypeAttr (NXDL data type), 153
deflection_angle (field), 241
deflector (base class), see NXdeflector, 472
defocus (field), 543
delay (field), 221
delay_difference (field), 590
density (field), 194, 233, 289, 296, 462
depends on (field attribute), 27
depends_on (field attribute), 180, 181, 218, 219, 314, 369, 370
depolarization (field), 491
depth (field), 241
description (field attribute), 276–282, 480, 533
description (file attribute), 276
design (field), 547, 586
design principles, 4
det_module (field), 277
details (field), 344
detection_gas_path (field), 206
detection_rate (field), 446
detector (base class), see NXdetector, 203
detector_distance (field), 522
detector_faces (field), 266
detector_group (base class), see NXdetector_group, 215
detector_identifier (field), 504, 511
detector_module (base class), see NXdetector_module, 217
detector_number (field), 197, 204, 401, 403
detector_readout_time (field), 210, 367
detector_type (field), 487, 539
diameter (field), 208, 271, 428, 508
dictionary of terms, 13
diffraction_order (field), 241
dim (NXDL element), 68
dimension, 20, 52
dimension scales, 53, 55–57
fasted varying, 44, 56, 218
slowest varying, 44, 218
storage order, 44
dimension scale, 24, 51
dimensions (NXDL element), 68, 142
dimensionsType (NXDL data type), 153
direction, see vector (field attribute)
direction (field attribute), 288, 289
DIRECTION (field), 180
direction (field), 302
directtof (application definition), see NXdirecttof, 350
disk_chopper (base class), see NXdisk_chopper, 219
distance_derived (field), 365
distance_to_detector (field), 182
distances (field), 315
distortion (base class), see NXdistortion, 473
distribution (field attribute), 201
divergence_x (field), 189
divergence_x_minus (field), 184
divergence_x_plus (field), 184
divergence_y (field), 190
divergence_y_minus (field), 185
divergence_y_plus (field), 184
doc (NXDL element), 68, 143
docType (NXDL data type), 155
documentation editor, 629
doi (field), 188
download location, see NAPI installation
dql (field), 339
dqw (field), 339
drain_current (field), 534, 539
duration (field), 224, 252, 309, 324, 403, 406, 491, 558, 559, 567, 568
duty_cycle (field), 241
dynamic_phi_list (field), 595
dynamic_q_list (field), 595
dynamic_roi_map (field), 594

e
ebeam_column (base class), see NXebeam_column, 474
electric_field (field), 397
efficiency (field), 212, 260, 272
ei (field), 397
electronanalyser (base class), see NEXlectronanalyser, 476
electrostatic_kicker (base class), see NEXelectrostatic_kicker, 479
element_names (field), 580
ellipsometry (application definition), see NEXellipsometry, 480
ellipsometry_type (field), 483
e (field), 317, 440, 483, 500
emittance_x (field), 305
emittance_y (field), 305
emitter_material (field), 475
emitter_type (field), 475, 514
e (field), 392, 398
end_time (field), 224, 259, 309, 324, 362, 375, 378, 380, 389, 394, 408, 411, 439, 467, 499, 504, 510, 558, 567, 588
end_time_estimated (field), 362
endnote (field), 188
energies (field), 329
energy (field), 231, 247, 262, 305, 328, 350, 352, 353, 390, 394, 395, 417, 419, 468
energy_error (field), 262
energy_errors (field), 262
energy_interval (field), 508
energy_resolution (field), 477, 537, 538
energy_scan_mode (field), 508, 539
energy_transfer (field), 178
energypedispersion (base class), see NEXenergypedispersion, 507
entering (field), 277
entrance_slit_setting (field), 329
entrance_slit_shape (field), 328
entrance_slit_size (field), 329
entry (base class), see NXEntry, 223
entry (group attribute), 328, 355, 384, 391, 416, 419
entry_identifier (field), 223, 309, 324, 558, 567, 588
entry_identifier_uuid (field), 223, 588
enumeration, 48
enumeration (NXDL element), 143
enumerationType (NXDL data type), 155
environment (base class), see NEXenvironment, 227

EPICS
  instrument examples, 122
equipment_component (field attribute), 314
error (field), 390
errors (field), 202
eulerian_cradle, 27, 323, 423
evaporation_idIncluded (field), 447
even_layer_density (field), 254
even_layer_material (field), 254
event_data (base class), see NXevent_data, 228
event_data_em (base class), see NXevent_data_em, 509
event_data_em_set (base class), see NXevent_data_em_set, 512
event_id (field), 229
event_identifier (field), 504, 510
event_index (field), 229, 560
event_pixel_id (field), 560
event_time_of_flight (field), 560
event_time_offset (field), 229
event_time_zero (field), 229
event_type (field), 504, 510
examples
  NeXus file, 5
  NeXus file; minimal, 6
experiment_description (field), 223, 309, 324, 438, 482, 499
experiment_identifier (field), 223, 309, 324, 438, 482, 499, 558, 567
experiments (field), 276
extends (NXDL attribute), 68
extent (field), 179, 468, 593
external_DAC (field), 291
external_field_brief (field), 299
external_material (field), 244, 254
extractor_current (field), 459
extractor_voltage (field), 459

F
fabrication (field), 238
faces (field), 266
facility_user_id (field), 317, 325, 562, 571
FAQ, 76
fast_axes (field), 477, 538
fast_pixel_direction (field), 218, 369
fax_number (field), 317
FDL, 630
features (field), 224
fermi_chopper (base class), see NEXfermi_chopper, 230
field, 4, 18
  HDF, 57
field (NXDL element), 68, 146
field attribute, 4, 15, 19
field_of_view (field), 443, 543
FIELDNAME_errors (field), 201
fileType (NXDL data type), 156
figure_of_merit (field), 584
file
  read and write, 13
  validate, 620
file attribute, 20
file format, 57
  HDF, 57
file_name (field), 264
file_name (file attribute), 286
file_time (file attribute), 286
file_update_time (file attribute), 286
filenames (field), 355, 392
filter (base class), see NXfilter, 232
final_beam_divergence (field), 180
final_energy (field), 178
final_polarization (field), 179
final_polarization_stokes (field), 179
final_wavelength (field), 179
final_wavelength_spread (field), 179
find the default plottable data, 49, 200, 223, 286, 309
firmware (field), 483
first_good (field attribute), 201
first_point_for_fit (field attribute), 592
fit_function (field), 457
fixed_energy (field), 389
fixed_revolution (field), 488
fixed_slit (field), 489
flags (field), 277
flatfield (field), 208, 366
flatfield_applied (field), 208, 365
flatfield_error (field), 366
flatfield_errors (field), 208, 366
flexible_name, 139
flight_path_length (field), 443
flip_current (field), 237
flip_turns (field), 237
flipper (base class), see NXflipper, 236
floating-point numbers, 47
fluo (application definition), see NXfluo, 351
flux (field), 180, 305, 371
flux (group attribute), 370
flux_integrated (field), 371
flyback_time (field), 555
focal_size (field), 186
focus_parameters (field), 238
focus_type (field), 320
focussing_probes (field), 483
folder, see group
format unification, 12
four-circle diffractometer, 27, 37, 323, 423
frame_average (field), 589
frame_start_number (field), 207, 421
frame_sum (field), 589
frame_time (field), 210, 367, 594
frequency (field attribute), 204
frequency (field), 190, 305, 560, 569
fresnel_zone_plate (base class), see NXfresnel_zone_plate, 238
G
  g2 (field), 589
g2_derr (field), 590
g2_err_from_two_time_corr_func (field), 592
  g2_err_from_two_time_corr_funcpartials (field), 593
  g2_from_two_time_corr_func (field), 591
  g2_from_two_time_corr_funcpartials (field), 592
  G2_unnormalized (field), 590
gain_setting (field), 210, 367
gap (field), 247
gas (field), 320
gas_pressure (field), 206, 320, 541
GDA (data acquisition software), 621
group, 4, 18
  HDF, 57
group (NXDL element), 68, 146
group attribute, 4, 19
group_index (field), 216, 364
group_names (field), 216, 364
group_parent (field), 216, 364
group_type (field), 216
groupType (NXDL data type), 160
guide (base class), see NXguide, 243
guide_current (field), 237
guide_turns (field), 237
Gumtree (data analysis software), 621
H
  h (field), 276
  h5py
    code examples, 86
  h5py_version (file attribute), 286
h5wasm	h5tools, 622
h5web	h5tools, 622
harmonic (field), 247
HDF
file format, 57
  tools, 622
HDF4, 626
HDF5, 626
HDF5_Version (file attribute), 286
HDF_version (file attribute), 286
HDFExplorer
tools, 622
HDFview
tools, 622
heater_power (field), 534
height (field), 231, 318
hierarchy, 4, 17, 18, 33, 35, 65, 66, 72, 619
high_trip_value (field), 298
hit_multiplicity (field), 447
hit_positions (field), 446
hit_rate (field), 524
holder (field), 562, 571

I

I (field), 337
I_axes (group attribute), 334
ibeam_column (base class), see NXibeam_column, 513
id (field), 276
identifier (field), 536, 562, 571
Idev (field), 338
IDF_Version (file attribute), 223, 309
IDL (data analysis software), 621
IGOR Pro (data analysis software), 621
image_id (field), 516, 522, 527
image_key (field), 209, 409
image_set_em_adf (base class), see NXimage_set_em_adf, 515
image_set_em_bf (base class), see NXimage_set_em_bf, 517
image_set_em_bse (base class), see NXimage_set_em_bse, 517
image_set_em_chamber (base class), see NXimage_set_em_chamber, 518
image_set_em_df (base class), see NXimage_set_em_df, 519
image_set_em_diffrac (base class), see NXimage_set_em_diffrac, 519
image_set_em_ecci (base class), see NXimage_set_em_ecci, 520
image_set_em_kikuchi (base class), see NXimage_set_em_kikuchi, 520
image_set_em_ronchigram (base class), see NXimage_set_em_ronchigram, 526
image_set_em_se (base class), see NXimage_set_em_se, 526
images, 47
incident_angle (field), 244, 254
incident_beam_divergence (field), 179, 468
incident_beam_energy (field), 468
incident_beam_size (field), 371
incident_energy (field), 178, 538, 593
incident_energy_spread (field), 468, 538, 593
incident_polarisation_stokes (field), 372
incident_polarization (field), 179, 538
incident_polarization_stokes (field), 179, 372
incident_polarization_type (field), 593
incident_wavelength (field), 178, 343, 370
incident_wavelength_spread (field), 179, 344, 371
incident_wavelength_weight (field), 370
incident_wavelength_weights (field), 178, 370
index (group attribute), 324
index (NXDL attribute), 68
indirecttof (application definition), see NXindirecttof, 353
ingestion, 619
initial_radius (field), 445
input (field), 206
insertion_device (base class), see NXinsertion_device, 246
inspection, 619
installation, see NAPI installation
instrument (base class), see NXinstrument, 249
instrument definitions, 8
instrument_name (field), 443, 503
int_prf (field), 281
int_prf_errors (field), 281
int_prf_var (field), 281
int_sum (field), 281
int_sum_errors (field), 282
int_sum_var (field), 282
integers, 46
integral (field), 259, 360, 378, 382, 412, 422
integral_counts (field), 404
intensity (field), 516, 521, 527, 544
intensity_threshold (field), 488
interaction_vol_em (base class), see NXinteraction_vol_em, 528
interior_atmosphere (field), 242, 244, 254
interpretation (field attribute), 468
introduction, 3
ion (base class), see NXion, 529
ion_energy_profile (field), 514
ion_type (field), 530
iqproc (application definition), see NXiqproc, 354
is_cylindrical (field), 194
ISAN (data analysis software), 621
ISO8601 (data type), 165
isotope_vector (field), 452, 530
issue reporting, 613
iupac_line_names (field), 580
J
jupyterlab-h5web
  tools, 622

K
k (field), 247, 276
kappa (field), 425
ki_over_kf_scaling (field), 389
Klosowski, Przemysław, 12, 626
Könnecke, Mark, 12, 627

L
l (field), 276
label (field), 544
lambda (field), 346
LAMP (data analysis software), 621
last_fill (field), 306
last_good (field attribute), 201
last_process (field), 457, 473, 554
laue_group (field), 523
laueof (application definition), see NXlaueof, 356
layer_structure (field), 489
layer_thickness (field), 242, 254
layout (field), 207
length (field), 247, 318, 341
lens_em (base class), see NXlens_em, 531
lens_geometry (field), 320
lens_length (field), 320
lens_material (field), 320
lens_mode (field), 328
lens_thickness (field), 320
lexicography, 13
LGPL, 630
license, 630
line_time (field), 555
link, 4, 20, 53, 78, 170, 199
  external file, 22, 23
link (NXDL element), 146
link target, 146
link target (internal attribute), 20
link, target, attribute, 20
linkType (NXDL data type), 162
local_name (field attribute), 206, 207
local_name (field), 206
location (field), 443, 503
log (base class), see NXLog, 251
long_name (field attribute), 201, 202, 204, 205, 516, 521,
  522, 527, 579–581
long_name (group attribute), 450, 451, 516, 579, 581
low_trip_value (field), 298
low-level file format, see file format
lp (field), 282
LRMECS

instrument examples, 119

M
m_value (field), 234, 244, 254
Mac OS X, see NAPI installation
magnetic_field (field), 184, 288, 326
magnetic_kicker (base class), see NXmagnetic_kicker, 532
magnetic_wavelength (field), 247
magnification (field), 459, 543
mailing lists, 612
manipulator (base class), see NXmanipulator, 534
Mantid (data analysis software), 621
manual source, 629
manufacturer (base class), see NXmanufacturer, 535
manufacturer (field), 186
manufacturer_model (field), 472
manufacturer_name (field), 444, 472, 532
mask (group attribute), 334
Mask_indices (group attribute), 334
mask_material (field), 238
mask_thickness (field), 238
mass (field), 289, 295
mass_to_charge (field), 448
mass_to_charge_range (field), 452, 530
material (field), 174, 486
MATLAB, 621
  code examples, 106
max_gap (field), 489
maximum_incident_angle (field), 186
maximum_number_of_atoms_per_molecular_ion
  (field), 451
maximum_value (field), 252, 558, 559, 567, 568
McStas, 31, 32
mean_angular_deviation (field), 523
measured_data (field), 490
measurement (field), 298
medium (field), 490
medium_refractive_indices (field), 490
metadata, 25, 64, 65, 72, 76, 142
method (field), 501
Microsoft Windows, see NAPI installation
miller_indices (field), 524
mime_type (group attribute), 310
min_bands (field), 522
min_intensity (field), 488
minimum_value (field), 252, 558, 559, 567, 568
mirror (base class), see NXmirror, 253
mode (field), 259, 306, 352, 358, 360, 376, 378, 382, 385,
  398, 401, 404, 407, 417, 422, 459, 522, 524,
  538, 562, 571
model (field), 297, 444, 483, 532, 536
moderator (base class), see NXmoderator, 256
module_offset (field), 218, 369
momentum_resolution (field), 477
monitor, 48
monitor (base class), see NXmonitor, 258
monochromator (base class), see NXmonochromator, 261
monopd (application definition), see NXmonopd, 359
mosaic_horizontal (field), 194
mosaic_vertical (field), 194
motivation, 3, see dictionary of terms, see exchange format, see format unification, see plotting, 12, 24
mpes (application definition), see NXmpes, 536
multi-dimensional data, 52
multi-modal data, 170
mx (application definition), see NXmx, 361

N
name (field attribute), 333
name (group attribute), 346
name (NXDL attribute), 68
naming convention, 40
NAPI, 4, 13, 14, 78, 598, 619, 626
bypassing, 57
c, 600
c++, 600
core, 599
f77, 600
f90, 600
IDL, 606
installation, 614
installation; download location, 615
installation; Mac OS X, 616
installation; RPM, 615
installation; source distribution, 616
installation; Windows, 616
java, 601
nature (field), 404, 406, 562, 571
ndattribute, 126
Nelson, Mitchell, 12
NeXpy, 7
NeXpy (data analysis software), 621
NeXus Application Programming Interface, see NAPI
NeXus Constructor, 620
NeXus Definition Language, see NXDL
NeXus International Advisory Committee, see NIAC

NeXus link, 20, 20, 23
NeXus webpage, 611
NeXus_version (file attribute), 286
NIAC, 10, 611, 611, 626
nominal (field), 259
nonNegativeUnbounded (NXDL data type), 165
note (base class), see NXnote, 263
notes (field), 558, 567
num (field), 318
number (field), 231
number_of_bunches (field), 305
number_of_cycles (field), 211
number_of_frames_averaged (field), 527
number_of_ion_types (field), 451
number_of_lenses (field), 320
number_of_reflectors (field), 524
number_of_runs (field), 490
number_sections (field), 244
numbers, see integers, see floating-point numbers
NX
used as NX class prefix, 23, 40
NX_ANGLE (units type), 166
NX_ANY (units type), 166
NX_AREA (units type), 166
NX_BINARY (data type), 165
NX_BOOLEAN (data type), 165
NX_CCOMPLEX (data type), 165
NX_CHAR (data type), 165
NX_CHARGE (units type), 166
NX_CLASS (file attribute), 286
NX_COMPLEX (data type), 165
NX_COUNT (units type), 166
NX_CROSS_SECTION (units type), 166
NX_CURRENT (units type), 166
NX_DATE_TIME (data type), 165
NX_DIMENSIONLESS (units type), 166
NX_EMITTANCE (units type), 166
NX_ENERGY (units type), 167
NX_FLOAT (data type), 165
NX_FLUX (units type), 167
NX_FREQUENCY (units type), 167
NX_INT (data type), 166
NX_LENGTH (units type), 167
NX_MASS (units type), 167
NX_MASS_DENSITY (units type), 167
NX_MOLECULAR_WEIGHT (units type), 167
NX_NUMBER (data type), 166
NX_PCOMPLEX (data type), 166
NX_PER_AREA (units type), 167
NX_PER_LENGTH (units type), 167
NX_PERIOD (units type), 167
NX_POSINT (data type), 166
NX_POWER (units type), 167
NX_PRESSURE (units type), 167

Index

nexus, Release v2022.07

639
NX_PULSES (units type), 167
NX_QUATERNION (data type), 166
NX_SCATTERING_LENGTH_DENSITY (units type), 168
NX_SOLID_ANGLE (units type), 168
NX_TEMPERATURE (units type), 168
NX_TIME (units type), 168
NX_TIME_OF_FLIGHT (units type), 168
NX_TRANSFORMATION (units type), 168
NX_UINT (data type), 166
NX_UNITLESS (units type), 168
NX_VOLTAGE (units type), 168
NX_VOLUME (units type), 168
NX_WAVELENGTH (units type), 168
NX_WAVENUMBER (units type), 169
NX_aberration (base class), 435
used in base class, 465
NX_aperture (base class), 173
used in application definition, 332, 481, 537, 558, 566
used in base class, 249, 459, 507
NX_aperture_em (base class), 436
used in application definition, 438
used in base class, 475, 514
NX_apm (application definition), 437
NX_archive (application definition), 323
NX_arpes (application definition), 327
NX_attenuator (base class), 175
used in application definition, 362, 430, 558, 566
used in base class, 249
NX_beam (base class), 177
used in application definition, 362, 467, 537, 588
used in base class, 249, 288, 462, 475, 514, 545
NX_beam_stop (base class), 182
used in base class, 249
NX_bending_magnet (base class), 184
used in base class, 249
nxbrowse, 16
nxbrowse (utility), 619
NX_calibration (base class), 457
used in application definition, 537
NX_canSAS (application definition), 330
NX_canSAS (applications)
dQ1, 340
dQw, 339
I, 337
Idev, 338
Q, 334
Qdev, 339
resolutions, 336
SAS_Aperture, 341
SAS_Collimation, 341
SAS_data, 333
SAS_detector, 341
SAS_entry, 332
SAS_instrument, 341
SAS_note, 345
SAS_process, 345
SAS_process_note, 345
SAS_sample, 344
SAS_source, 342
SAS_transmission_spectrum, 346
Tdev, 347
NX_capillary (base class), 186
used in base class, 249
NX_chamber (base class), 458
used in application definition, 438
NX_cite (base class), 187
NX_class (attribute), 40
NX_collection (base class), 188
used in application definition, 332, 362, 389, 438, 467, 558, 566
used in base class, 203, 223, 249, 308, 521, 544, 545
NX_collection_column (base class), 459
used in application definition, 537
used in base class, 477
NX_collector (base class), 189
used in application definition, 332, 380, 384
used in base class, 249
NX_container (base class), 460
nx_convert (utility), 619
NX_coordinate_system_set (base class), 463
used in application definition, 438, 499
NX_corrector_cs (base class), 464
used in base class, 475
NX_crystal (base class), 191
used in application definition, 359, 397, 558, 566
used in base class, 249, 262
NX_csg (base class), 466
used in base class, 466, 577
NX_cxi_pycro (application definition), 467
NX_cylindricalGeometry (base class), 197
used in base class, 182, 203
NX_data (base class), 5, 56, 198
<table>
<thead>
<tr>
<th>NXdeflector (base class), 472</th>
<th>used in base class, 459, 477, 507, 584</th>
</tr>
</thead>
<tbody>
<tr>
<td>used in base class, 249, 477</td>
<td></td>
</tr>
<tr>
<td>NXdetector_group (base class), 215</td>
<td>used in application definition, 362</td>
</tr>
<tr>
<td>used in base class, 249</td>
<td></td>
</tr>
<tr>
<td>NXdetector_module (base class), 217</td>
<td>used in application definition, 362</td>
</tr>
<tr>
<td>used in base class, 203</td>
<td></td>
</tr>
<tr>
<td>nxdir (utility), 619</td>
<td></td>
</tr>
<tr>
<td>NXdirectttof (application definition), 350</td>
<td></td>
</tr>
<tr>
<td>NXdisk_chopper (base class), 219</td>
<td>used in application definition, 350, 377, 558, 566</td>
</tr>
<tr>
<td>used in base class, 249</td>
<td></td>
</tr>
<tr>
<td>NXDL (base class), 473</td>
<td></td>
</tr>
<tr>
<td>NXDL, 23, 77, 137, 141, 141</td>
<td></td>
</tr>
<tr>
<td>NXDL elements, 142</td>
<td></td>
</tr>
<tr>
<td>NXDL template file, 68</td>
<td></td>
</tr>
<tr>
<td>nxdl_to_hd5.py, 620</td>
<td></td>
</tr>
<tr>
<td>NXebeam_column (base class), 474</td>
<td>used in application definition, 499</td>
</tr>
<tr>
<td>used in base class, 510</td>
<td></td>
</tr>
<tr>
<td>NXelectronanlyser (base class), 476</td>
<td>used in application definition, 537</td>
</tr>
<tr>
<td>used in application definition, 537</td>
<td></td>
</tr>
<tr>
<td>NXelectrostatic_kicker (base class), 479</td>
<td></td>
</tr>
<tr>
<td>NXellipsometry (application definition), 480</td>
<td></td>
</tr>
<tr>
<td>NXem (application definition), 495</td>
<td></td>
</tr>
<tr>
<td>NXenergydispersion (base class), 507</td>
<td>used in application definition, 537</td>
</tr>
<tr>
<td>used in application definition, 537</td>
<td></td>
</tr>
<tr>
<td>used in base class, 286</td>
<td></td>
</tr>
<tr>
<td>NXenvironment (base class), 227</td>
<td>used in application definition, 481</td>
</tr>
<tr>
<td>used in base class, 288</td>
<td></td>
</tr>
<tr>
<td>NXevent_data (base class), 228</td>
<td>used in application definition, 558</td>
</tr>
<tr>
<td>used in base class, 249</td>
<td></td>
</tr>
<tr>
<td>NXevent_data_em (base class), 509</td>
<td>used in application definition, 499</td>
</tr>
<tr>
<td>used in base class, 513</td>
<td></td>
</tr>
<tr>
<td>NXevent_data_em_set (base class), 512</td>
<td>used in application definition, 499</td>
</tr>
<tr>
<td>NXfermi_chopper (base class), 230</td>
<td>used in application definition, 350, 389, 566</td>
</tr>
<tr>
<td>used in base class, 249</td>
<td></td>
</tr>
<tr>
<td>NXfilter (base class), 232</td>
<td>used in base class, 249</td>
</tr>
<tr>
<td>NXflipper (base class), 236</td>
<td>used in base class, 249</td>
</tr>
<tr>
<td>NXfluo (application definition), 351</td>
<td></td>
</tr>
<tr>
<td>NXfresnel_zone_plate (base class), 238</td>
<td></td>
</tr>
<tr>
<td>NXgeometry (base class), 239</td>
<td>used in application definition, 380, 384, 558, 566</td>
</tr>
<tr>
<td>NXgrating (base class), 241</td>
<td>used in application definition, 481</td>
</tr>
<tr>
<td>used in base class, 262</td>
<td></td>
</tr>
<tr>
<td>NXguide (base class), 243</td>
<td></td>
</tr>
<tr>
<td>used in base class, 249</td>
<td></td>
</tr>
<tr>
<td>NXibeam_column (base class), 513</td>
<td>used in application definition, 499</td>
</tr>
<tr>
<td>used in base class, 510</td>
<td></td>
</tr>
<tr>
<td>NXimage_set_em_adf (base class), 515</td>
<td>used in application definition, 499</td>
</tr>
<tr>
<td>used in base class, 510</td>
<td></td>
</tr>
<tr>
<td>NXimage_set_em_bf (base class), 517</td>
<td>used in application definition, 499</td>
</tr>
<tr>
<td>used in base class, 510</td>
<td></td>
</tr>
<tr>
<td>NXimage_set_em_bse (base class), 517</td>
<td>used in application definition, 499</td>
</tr>
<tr>
<td>used in base class, 510</td>
<td></td>
</tr>
<tr>
<td>NXimage_set_em_chamber (base class), 518</td>
<td>used in application definition, 499</td>
</tr>
<tr>
<td>used in base class, 510</td>
<td></td>
</tr>
<tr>
<td>NXimage_set_em_df (base class), 519</td>
<td>used in application definition, 499</td>
</tr>
<tr>
<td>used in base class, 510</td>
<td></td>
</tr>
<tr>
<td>NXimage_set_em_diffrac (base class), 519</td>
<td>used in application definition, 499</td>
</tr>
<tr>
<td>used in base class, 510</td>
<td></td>
</tr>
<tr>
<td>NXimage_set_em_ecci (base class), 520</td>
<td>used in application definition, 499</td>
</tr>
<tr>
<td>used in base class, 510</td>
<td></td>
</tr>
<tr>
<td>NXimage_set_em_kikuchi (base class), 520</td>
<td>used in application definition, 499</td>
</tr>
<tr>
<td>used in base class, 510</td>
<td></td>
</tr>
<tr>
<td>NXimage_set_em_ronchigram (base class), 526</td>
<td>used in application definition, 499</td>
</tr>
<tr>
<td>used in base class, 510</td>
<td></td>
</tr>
<tr>
<td>NXimage_set_em_se (base class), 526</td>
<td></td>
</tr>
</tbody>
</table>
used in application definition, 499
used in base class, 510
NXindirecttof (application definition), 353
nxingest (utility), 619
NXinsertion_device (base class), 246
used in base class, 249
NXinstrument (base class), 6, 249
used in base class, 223, 308
NXinteraction_vol_em (base class), 528
NXion (base class), 529
used in application definition, 438
used in base class, 514, 579
NXiqproc (application definition), 354
NXlauetof (application definition), 356
NXlens_em (base class), 531
used in application definition, 438
used in base class, 459, 465, 475, 477, 507, 514, 584
NXlog (base class), 251
used in application definition, 558, 566
used in base class, 189, 191, 233, 256, 259, 288, 297, 479, 533, 549, 556, 576, 582
NXmagnetic_kicker (base class), 532
NXmanipulator (base class), 534
used in application definition, 537
NXmanufacturer (base class), 535
used in application definition, 438, 499
used in base class, 437, 457, 465, 475, 514, 531, 545, 550, 586
NXmirror (base class), 253
used in base class, 249
NXmoderator (base class), 256
used in application definition, 558, 566
used in base class, 249
NXmonitor (base class), 258
used in application definition, 351, 357, 359, 375, 377, 380, 384, 388, 394, 397, 400, 403, 406, 408, 411, 416, 420, 438, 467, 499, 558, 566
used in base class, 223, 308
NXmonochromator (base class), 261
used in application definition, 328, 351, 353, 375, 380, 394, 416, 420, 481
used in base class, 249
NXmonopd (application definition), 359
NXmpes (application definition), 536
NXmx (application definition), 361
NXnote (base class), 263
used in application definition, 332, 438, 499, 537, 558, 566, 588
used in base class, 174, 203, 223, 227, 275, 303, 308, 320
NXobject (base class), 264
NXoff_geometry (base class), 265
NXoptical_system_em (base class), 543
used in application definition, 499
used in base class, 510, 527
NXorientation (base class), 266
used in application definition, 558, 566
used in base class, 240, 297
NXparameters (base class), 267
used in application definition, 354, 391, 414, 419
used in base class, 223, 308
NXpdb (base class), 268
NXpeak (base class), 544
used in application definition, 438
used in base class, 579
NXpinhole (base class), 270
NXplot (utility), 620
NXpolarizer (base class), 271
used in application definition, 558, 566
used in base class, 249
NXpositioner (base class), 272
used in application definition, 558, 566, 588
used in base class, 249, 288, 534, 586
NXprocess, 72
NXprocess (base class), 274
used in application definition, 332, 354, 391, 414, 419, 438, 481, 537, 588
used in base class, 223, 308, 515, 517–521, 526, 527, 529, 577–579
NXpulser_apm (base class), 545
used in application definition, 438
NXpump (base class), 547
used in application definition, 438, 499
NXquadric (base class), 547
used in base class, 577
NXquadrupole_magnet (base class), 549
NXreflection (base class), 275
NXreflectron (base class), 550
used in application definition, 438
NXrefscan (application definition), 375
NXreftof (application definition), 377
NXregistration (base class), 554
NXroot (base class), 285
attributes, 20
NXsample (base class), 6, 287
used in base class, 223, 308
NXsample_component (base class), 294
used in base class, 288
NXsas, 625
NXsas (application definition), 380
NXsasof (application definition), 383
NXscan (application definition), 387
NXscanbox_em (base class), 555
used in application definition, 499
used in base class, 510
NXsensor (base class), 297
used in application definition, 481
used in base class, 227, 233, 475, 514
NXseparator (base class), 556
NXshape (base class), 300
used in application definition, 380, 384, 558, 566
used in base class, 191, 240, 241, 254, 462
NXslit (base class), 302
used in application definition, 481
NXsnevent (application definition), 557
NXsnshisto (application definition), 566
NXsolenoind_magnet (base class), 575
NXsolid_geometry (base class), 576
NXsource (base class), 303
used in application definition, 324, 328, 332, 351, 354, 359, 362, 375, 380, 384, 391, 394, 397, 408, 411, 414, 416, 420, 427, 467, 481, 537, 558, 566
used in base class, 249, 475, 514, 545
NXspe (application definition), 389
NXspectrum_set_em_auger (base class), 577
used in application definition, 499
used in base class, 510
NXspectrum_set_em_cathodolum (base class), 577
used in application definition, 499
used in base class, 510
NXspectrum_set_em_eels (base class), 578
used in application definition, 499
used in base class, 510
NXspectrum_set_em_xray (base class), 579
used in application definition, 499
used in base class, 510
NXspin_rotator (base class), 582
NXspindispersion (base class), 583
used in base class, 477
NXsqom (application definition), 391
NXstage_lab (base class), 585
used in application definition, 438
used in base class, 475
NXstxm (application definition), 393
NXsubentry (base class), 308
used in application definition, 481
used in base class, 223
nxsummary, 619
NXtas (application definition), 397
NXtofnpd (application definition), 400
NXtofraw (application definition), 402
NXtofsingle (application definition), 405
NXtomo (application definition), 408
NXtomophase (application definition), 411
NXtomoproc (application definition), 414
NXtransformations (base class), 312
used in application definition, 362, 467, 481
nxtranslate (utility), 620
NXtranslation (base class), 315
used in application definition, 558, 566
used in base class, 240
NXuser (base class), 316
used in application definition, 324, 400, 403, 406, 438, 481, 499, 558, 566
used in base class, 223, 308, 510
nxvalidate, 76
NXvelocity_selector (base class), 317
used in base class, 249, 262
NXsas (application definition), 416
NXxasproc (application definition), 418
NXxbasis (application definition), 420
NXxeuler (application definition), 423
NXxkappa (application definition), 425
NXxlaue (application definition), 426
NXxlaueplate (application definition), 427
NXxnb (application definition), 428
NXxpcs (application definition), 587
NXxraylens (base class), 319
used in base class, 249
NXxrot (application definition), 430

O

object (base class), see NXobject, 264
observed_frame (field), 278
observed_frame_errors (field), 279
observed_frame_var (field), 279
observed_phi (field), 280
observed_phi_errors (field), 280
observed_phi_var (field), 280
observed_px_x (field), 279
observed_px_x_errors (field), 279
observed_px_x_var (field), 279
observed_px_y (field), 279
observed_px_y_errors (field), 279
observed_px_y_var (field), 279
observed_x (field), 280
observed_x_errors (field), 280
observed_x_var (field), 280
observed_y (field), 280
observed_y_errors (field), 281
observed_y_var (field), 280
odd_layer_density (field), 254
odd_layer_material (field), 254
off_geometry (base class), see NXoff_geometry, 265
offset (field attribute), 27, 180, 218, 229, 314, 369, 370, 415
offset (field), 202, 457
offset_units (field attribute), 218, 219, 314
OpenGENIE (data analysis software), 621
operation (field), 466
operation_mode (field), 440
optical_system_em (base class), see NXoptical_system_em, 543
ORCID (field), 317
orcid (field), 441, 483, 500
order (transformation), see depends on (field attribute)
order_no (field), 193
orientation (base class), see NXorientation, 266
orientation_angle (field), 487
orientation_matrix (field), 193, 234, 289, 295, 358, 398, 421
original_axis (field), 457
original_centre (field), 474
original_points (field), 474
Osborn, Raymond, 627
other_detector (field), 487
other_material (field), 487
outer_diameter (field), 238
outermost_zone_width (field), 238
overlaps (field), 282

P

packing_fraction (field), 462
pair_separation (field), 220
parameters (base class), see NXparameters, 267
parameters (field), 548
parent (field), 552
parent_mask (field), 552
partiality (field), 278
pass_energy (field), 329, 508, 539
path_length (field), 291
path_length_window (field), 291

pattern_center (field), 522
pattern_quality (field), 522
pdb (base class), see NXpdb, 268
peak (base class), see NXpeak, 544
peak_model (field), 544
peaks (field), 580
period (field), 241, 305
phase (field), 221, 247
phase_identifier (field), 523, 524
phase_name (field), 524
phi (field), 424, 426
photon_energy (field), 580
physical file format, see file format
pinhole (base class), see NXpinhole, 270
pitch (field), 342, 344
pixel_id (field), 560, 569
pixel_mask (field), 208, 366
pixel_mask_applied (field), 208, 366
pixel_time (field), 555
how to find data, 48
point_group (field), 291, 296
poison_depth (field), 257
poison_material (field), 257
polar (field), 389
polar_angle (field), 194, 205, 282, 354, 357, 360, 376, 378, 381, 385, 397, 398, 401, 403, 406, 424, 425, 429, 431, 560, 569
polar_width (field), 389
polarizer (base class), see NXpolarizer, 271
poles (field), 247
populated_elements (field attribute), 591
position (field), 544, 587
positioner (base class), see NXpositioner, 272
power (field), 247, 305, 546
pre_sample_flightpath (field), 225, 310, 400, 403, 406
precompiled executable, see NAPI installation
predicted_frame (field), 278
predicted_phi (field), 278
predicted_px_x (field), 278
predicted_px_y (field), 278
predicted_x (field), 278
predicted_y (field), 278
preparation_date (field), 290, 326, 441, 489, 502, 540
preset (field), 259, 352, 358, 360, 376, 378, 382, 385, 398, 401, 404, 407, 417, 422
pressure (field), 289, 326, 445
prf_cc (field), 282
primary (field attribute), 204, 205
probe (field), 304, 325, 352, 355, 360, 376, 380, 384,
    391, 394, 397, 409, 411, 414, 417, 421, 468,
    475, 537, 560, 569
process (base class), see NXprocess, 274
Processed Data, 72
profile (field), 371
program (field), 228, 275, 325, 355, 392, 414, 419, 439,
    446–452, 482, 499, 515, 580
program_name (field), 225, 310, 389
programs, 619
projection (field), 459, 538
protocol_name (field), 449
proton_charge (field), 558, 567
psi (field), 389
pulse_energy (field), 491, 546
pulse_fraction (field), 444, 545
pulse_frequency (field), 444, 545
pulse_height (field), 229
pulse_id (field), 447
pulse_mode (field), 444, 545
pulse_time (field), 560
pulse_width (field), 306
pulsed_voltage (field), 444, 545
pulser_apm (base class), see NXpulser_apm, 545
pulses_since_last_ion (field), 447
pump (base class), see NXpump, 547
punx (utility), 620
PyMCA (data analysis software), 622
Q
Q (field), 334
Q_indices (group attribute), 334
Qdev (field), 339
qh (field), 398
qk (field), 398
ql (field), 398
Qmean (field), 340
quadric (base class), see NXquadric, 547
quadrupole_magnet (base class), see
    NXquadrupole_magnet, 549
qx (field), 355, 392
qy (field), 356, 392
qz (field), 392
R
r_slit (field), 231
radiation (field), 342
radius (field), 221, 231, 318
range (field), 259
range_increment (field), 451
range_minmax (field), 451
rank, 15, 52, 57, 68
rank (NXDL attribute), 68
ratio (field), 221
raw (field), 539
raw_file (field), 415, 419
raw_frames (field), 558, 567
raw_time_of_flight (field), 204
raw_tof (field), 448
raw_value (field), 252, 273
rdeform_field (field), 474
read file, 15
real_time (field), 207
reconstructed_positions (field), 449
reference_plane (field), 180
reference_sample (field), 487
reference_wavelength (field), 487
reflection (field), 193, 272
reflection_id (field), 277
reflections (base class), see NXreflections, 275
reflectron (base class), see NXreflectron, 550
refscan (application definition), see NXrefscan, 375
reftof (application definition), see NXreftof, 377
region (base class), see NXregion, 551
region_origin (field), 329
region_size (field), 329
region_type (file attribute), 552
registration (base class), see NXregistration, 554
regular expression, 40
relative_molecular_mass (field), 289, 296, 462
release
    NeXus definitions, 616
    notes, 616
    process, 616
    tags, 137, 617
    versioning, 137, 617
release_date (field), 325
repository, 613, see NAPI installation
requested_pixel_time (field), 555
reserved prefixes, 42
    BLUESKY_, 42
    DECTRIS_, 42
    IDF_, 42
    NDAattr, 42
    NX, 42
    NX_, 42
    PDBX_, 42
    SAS_, 42
    SILX_, 42
reserved suffixes, 42
    end, 43
    errors, 43
    increment_set, 43
    indices, 43
    mask, 43
    set, 43
weights, 43
resolution (field), 524
resolutions (field attribute), 336
resolutions_description (field attribute), 336
revision (field), 225, 310, 324
revision history, 630
revolution (field), 488
Riedel, Richard, 626
roi (field), 527
role (field), 316, 325, 441, 501, 562, 571
roll (field), 342, 344
root (base class), see NXroot, 285
rotating_element (field), 488
rotation, 27
rotation (field), 555, 587
rotation_angle (field), 291, 360, 376, 378, 381, 385, 388, 390, 394, 397, 398, 409, 412, 424, 425, 429, 431
rotation_angle_step (field), 431
rotation_speed (field), 220, 231, 318, 350
RPM, see NAPI installation
rules, 3
HDF, 18, 139
HDF5, 41
naming, 23, 32, 40, 139
NeXus, 32
NX prefix, 23
XML, 139
run (field), 333
run_control (field), 298
run_cycle (field), 224, 310, 324
run_number (field), 403, 439, 558, 567
S
sample (base class), see NXsample, 287
sample_bias (field), 534, 539
sample_component (base class), see NXsample
sample_component (field), 290
sample_history (field), 441, 489, 501
sample_id (field), 325
sample_name (field), 489
sample_orientation (field), 289, 295
sample_temperature (field), 534, 539
sample_x (field), 395
sample_y (field), 395
sampled_fraction (field), 260
sas (application definition), see Nexus, 380
sastof (application definition), see Nexus, 383
saturation_value (field), 211, 367
scale (field), 553
scaling (field attribute), 415
scaling (field), 457
scaling_factor (field attribute), 252, 253, 338
scaling_factor (field), 202
scan (application definition), see Nexus, 387
scan_number (field), 588
scanbox_em (base class), see Nexus, 555
scattering_angle (field), 584
scattering_cross_section (field), 176
scattering_energy (field), 584
scattering_length_density (field), 290, 296
scattering_vector (field), 193
scheme (field), 459, 507, 538
Scientific Data Sets, see field
SDD (field), 341
SDS (Scientific Data Sets), see field
seblock (field), 390
segment_columns (field), 194
segment_gap (field), 194
segment_height (field), 194
segment_rows (field), 194
segment_thickness (field), 194
segment_width (field), 194
semi_convergence_angle (field), 543
sensor (base class), see NXsensor, 297
sensor_material (field), 211, 368
sensor_size (field), 329
sensor_thickness (field), 211, 368
separator (base class), see Nexus, 556
sequence_index (field), 264, 275
sequence_number (field), 207, 412
serial_number (field), 206, 444
set_Bfield_current (field), 556, 582
set_current (field), 480, 533, 549, 576
set_Efield_voltage (field), 556, 582
set_voltage (field), 480, 533
gsl (field), 398
gsu (field), 398
ShadowFactor (field), 340
shank_angle (field), 445
shape (base class), see Nexus, 300
shape (field), 301, 341, 381, 385, 560–562, 569–571
shermann_function (field), 584
short_name (field attribute), 249, 304, 363, 372, 477
short_name (field), 227, 297
short_title (field), 291, 442, 502
sigma_x (field), 305
sigma_y (field), 305
signal (field attribute), 201, 357, 421
signal (file attribute), 200
signal (group attribute), 212, 245, 333, 346, 382, 468, 469, 539, 541
signal attribute value, 49, 200
signal data, 19, 49
signal_amplitude (field), 444
silx (data analysis software), 621
situation (field), 290, 326, 540
Index 647

T (field), 346
T_axes (group attribute), 346
table (field), 318
tags, 137, 617
taper (field), 247
target (field), 584
target, attribute, 20
target_material (field), 305
target_preparation (field), 584
target_preparation_date (field), 584
target_value (field), 273
tas (application definition), see NXnias, 397

standing_voltage (field), 545
start (field attribute), 207, 221, 230, 252
start (field), 552
start_time (field), 207, 224, 259, 309, 324, 328, 350, 351, 353, 359, 362, 375, 378, 380, 384, 388, 394, 397, 400, 403, 406, 408, 411, 416, 421, 438, 467, 482, 499, 504, 510, 537, 558, 567, 588
static_q_list (field), 595
static_roi_map (field), 595
status (field), 176, 182, 233, 523
step_size (field), 521
stop_time (field), 207
storage_mode (field attribute), 589–593
strategies, 72
  simplest case(s), 73
stress_field (field), 288, 326
stride (field), 552
strings, 47
  arrays, 47
  fixed-length, 47
  variable-length, 47
stxm (application definition), see NXstxm, 393
stxm_scan_type (field), 394
subentry
  NXsubentry, 170
subentry (base class), see NXsubentry, 308
substrate_density (field), 242, 254
substrate_material (field), 234, 242, 244, 254
substrate_roughness (field), 234, 242, 244, 254
substrate_thickness (field), 234, 242, 244, 254
support_membrane_material (field), 239
support_membrane_thickness (field), 239
surface (field), 245
surface_indices (group attribute), 245
surface_type (field), 548
symbols (NXDL element), 146
symbolsType (NXDL data type), 162
symmetric (field), 320
symmetry (field), 474

T

size (field), 182, 301, 381, 385, 560–562, 569–571
slit (base class), see NXslit, 302
slit (field), 231
slit_angle (field), 220
slit_edges (field), 221
slit_height (field), 221
slit_length (field), 342
slits (field), 220
slot (field), 206
slow_axes (field), 478, 538
slow_pixel_direction (field), 218, 369
SNsbanking_file_name (field), 559, 568
SNsdetector_calibration_id (field), 559, 568
SNsgeometry_file_name (field), 560, 569
SNshisto (application definition), see NXSNShisto, 566
SNsmapping_file_name (field), 559, 568
SNstranslation_service (field), 560, 569
social_media_name (field), 441, 501
social_media_platform (field), 441, 501
soft_limit_max (field), 273
soft_limit_min (field), 273
software, 619, 620
solenoid_magnet (base class), see NXsolenoid_magnet, 575
solid_angle (field), 206
solid_geometry (base class), see NXsolid_geometry, 576
soller_angle (field), 189
source (base class), see NXsource, 303
source distribution, see NAPI installation
source_distance_x (field), 184
source_distance_y (field), 184
space_group (field), 193, 291, 296, 523
spatial_resolution (field), 477
spe (application definition), see NXspe, 389
spec2nexus, 622
spectral_resolution (field), 488
spectrum_set_em_auger (base class), see NXspectrum_set_em_auger, 577
spectrum_set_em_cathodolum (base class), see NXspectrum_set_em_cathodolum, 577
spectrum_set_em_eels (base class), see NXspectrum_set_em_eels, 578
spectrum_set_em_xray (base class), see NXspectrum_set_em_xray, 579
Sphinx (documentation generator), 629
spin_rotator (base class), see NXspin_rotator, 582
spindispersion (base class), see NXspindispersion, 583
spwidth (field), 318
sqom (application definition), see NXSQom, 391
stage_lab (base class), see NXstage_lab, 585
stage_type (field), 485

nexus, Release v2022.07
Tdev (field), 346
telephone_number (field), 316, 441, 483, 501
temperature (field), 194, 233, 257, 288, 326, 329, 344, 363, 390, 422, 540, 561, 570, 595
temperature_coefficient (field), 194
temperature_set (field), 595
template, see NXDL template file
term (field), 267, 345
thickness (field), 176, 193, 233, 237, 291, 344, 487, 502
threshold_energy (field), 211, 368
tiled
tools, 622
tilt_1 (field), 587
tilt_2 (field), 587
tilt_angle (field), 429
time (field attribute), 176, 306
time (field), 252, 558, 559, 567, 568
time_of_flight (field), 204, 260, 357, 358, 378, 385, 386, 401, 403, 404, 406, 407, 562, 569, 571
time_origin_location (field attribute), 591
time_per_channel (field), 329, 365
time_points (field), 490
time_zone (field), 363
timestamp (group attribute), 334, 346
timing (field), 480, 533
Tischler, Jonathan, 12, 627
tof_distance (field), 508
tofnpd (application definition), see NXtofnpd

tofraw (application definition), see NXtofraw
tofsingle (application definition), see NXtofsingle
tolerance (field), 273
tomo (application definition), see NXtomo
tomophase (application definition), see NXtomophase

tomoproc (application definition), see NXtomoproc

top_dead_center (field), 221
top_up (field), 306
total_counts (field), 558, 560, 567, 569
total_flux (field), 371
total_flux_integrated (field), 371
total_uncounted_counts (field), 558, 567
transform (field attribute), 415
transformation matrices, 26
transformation_type (field attribute), 27
transformation_type (field attribute), 180, 218, 313, 369
transformations (base class), see NXtransformations
translation (field), 468
transmission (field), 344
transmitting_material (field), 190, 231
tree structure, see hierarchy
trigger_dead_time (field), 210
trigger_delay_time (field), 210
trigger_delay_time_set (field), 210
trigger_internal_delay_time (field), 210
tutorial
WONI, 60
twist (field), 318
two_time_corr_func (field), 590
type, see data type

type (group attribute), 225, 440, 500
type (NXDL attribute), 68

U
ub_matrix (field), 289

UDunits, 47, 48

uncertainties (field attribute), 335, 338, 346

underload_value (field), 211, 368

Unidata UDunits, 47

unit category, 166

unit_cell (field), 193, 289, 358, 398, 421

unit_cell_a (field), 193, 234

unit_cell_ab (field), 289, 295, 523

unit_cell_alpha (field), 193, 234

unit_cell_alphabetagamma (field), 289, 295, 523

unit_cell_b (field), 193, 234

unit_cell_beta (field), 193, 234

unit_cell_c (field), 193, 234

unit_cell_class (field), 290, 296, 523

unit_cell_gamma (field), 193, 234

unit_cell_volume (field), 193, 234, 289, 295

units, 15, 19, 47, 140

units (NXDL attribute), 68

URL (field attribute), 267, 335, 337–340, 468, 469

URL(U) (field attribute), 267, 335, 337–340, 468, 469

url (field attribute), 482, 483

url (field), 188

url (group attribute), 482

usage (field), 192

use_of, 170

user (base class), see NXuser, 316

UTF-8, 47, 165

utilities, 619