nexus

Release v2024.02

NIAC, https://www.nexusformat.org

Feb 09, 2024

CONTENTS

https://www.nexusformat.org/

CHAPTER

ONE

NEXUS: USER MANUAL

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 underwent 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.

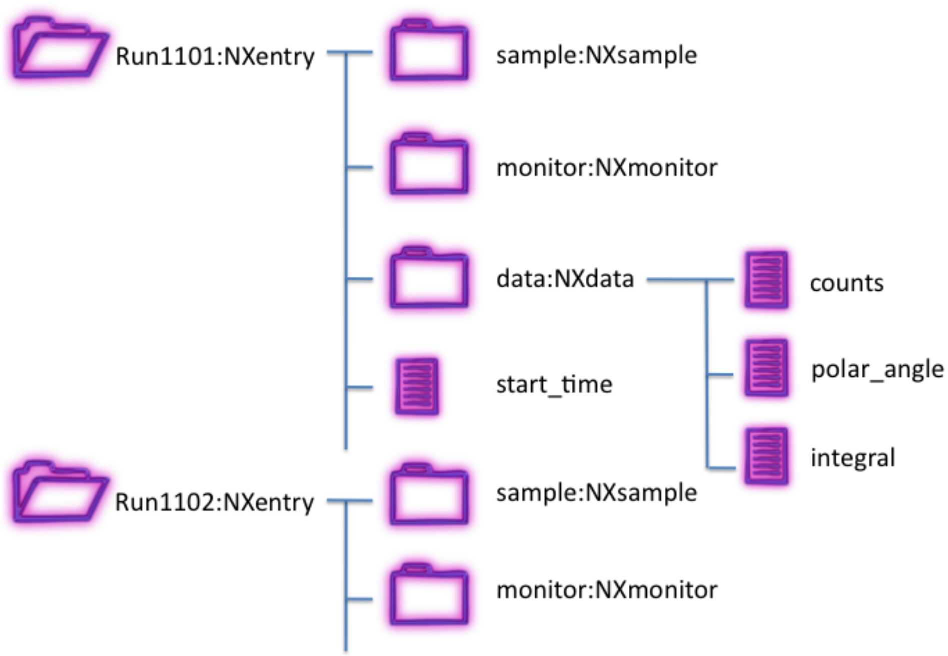
Links
Links are used to represent the same information in different places.

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



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`).

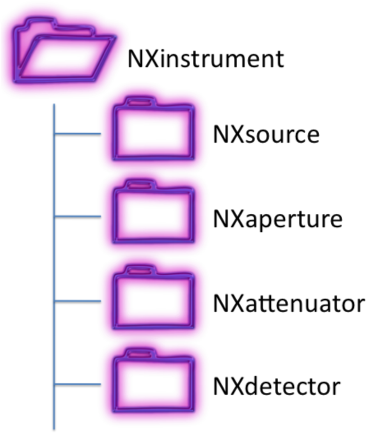
NXdata
: Each `NXentry` group may contain one or more `NXdata` groups. These groups contain the experimental results

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 `NXdata` group. That means it should contain the axis labels and titles as well as the data.

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

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

`NXinstrument` excerpt



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.

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.)

Example structure of a simple data file

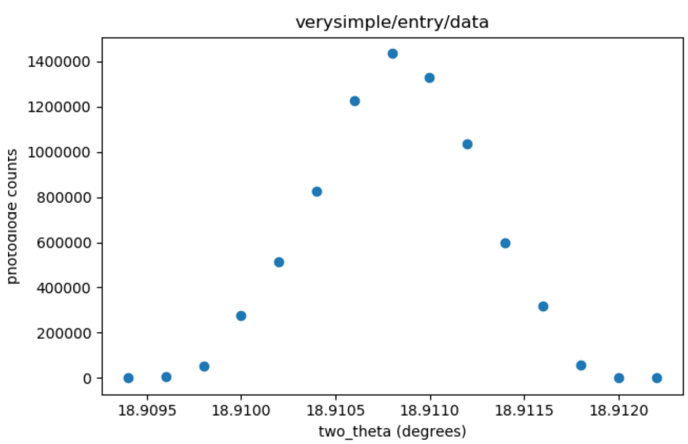
This illustrates the fact that the structure of NeXus files is extremely flexible. It can accommodate very complex instrumental information, if required, but it can also be used to store very simple data sets. Here is the structure of a very simple NeXus data file (`examples/verysimple.nx5`):

Structure of a very simple NeXus Data file

```
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

angle = [18.9094, 18.9096, 18.9098, 18.91,  18.9102,
         18.9104, 18.9106, 18.9108, 18.911, 18.9112,
         18.9114, 18.9116, 18.9118, 18.912, 18.9122]
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

```
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:

example of NeXus data file structure

```
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)

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

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 maintained 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.

```
#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][6] which are specified as a character string (`type="NX_CHAR"`[7]) 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

```
NXopen ('NXfile.nxs', NXACC_READ, &fileID);
  NXopengroup (fileID, "Scan", "NXentry");
    NXopengroup (fileID, "data", "NXdata");
      NXopendata (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 `nxbrowse` to view a data file.

Using `nxbrowse`

```
%> nxbrowse lrcs3701.nxs

NXBrowse 3.0.0. Copyright (C) 2000 R. Osborn, M. Koennecke, P. Klosowski
    NeXus_version = 1.3.3
    file_name = lrcs3701.nxs
    file_time = 2001-02-11 00:02:35-0600
    user = EAG/RO
NX> dir
  NX Group : Histogram1 (NXentry)
  NX Group : Histogram2 (NXentry)
NX> open Histogram1
NX/Histogram1> dir
  NX Data  : title[44] (NX_CHAR)
  NX Data  : analysis[7] (NX_CHAR)
  NX Data  : start_time[24] (NX_CHAR)
  NX Data  : end_time[24] (NX_CHAR)
  NX Data  : run_number (NX_INT32)
  NX Group : sample (NXsample)
  NX Group : LRMECS (NXinstrument)
  NX Group : monitor1 (NXmonitor)
  NX Group : monitor2 (NXmonitor)
  NX Group : data (NXdata)
NX/Histogram1> read title
  title[44] (NX_CHAR) = MgB2 PDOS 43.37g 8K 120meV E0@240Hz T0@120Hz
NX/Histogram1> open data
NX/Histogram1/data> dir
  NX Data  : title[44] (NX_CHAR)
  NX Data  : data[148,750] (NX_INT32)
```

(continues on next page)

(continued from previous page)

```
29   NX Data  : time_of_flight[751] (NX_FLOAT32)
30   NX Data  : polar_angle[148] (NX_FLOAT32)
31 NX/Histogram1/data> read time_of_flight
32   time_of_flight[751] (NX_FLOAT32) = [ 1900.000000 1902.000000 1904.000000 ...]
33     units = microseconds
34     long_name = Time-of-Flight [microseconds]
35 NX/Histogram1/data> read data
36   data[148,750] (NX_INT32) = [ 1 1 0 ...]
37     units = counts
38     signal = 1
39     long_name = Neutron Counts
40     axes = polar_angle:time_of_flight
41 NX/Histogram1/data> close
42 NX/Histogram1> close
43 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 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 x m x 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 x m x 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.

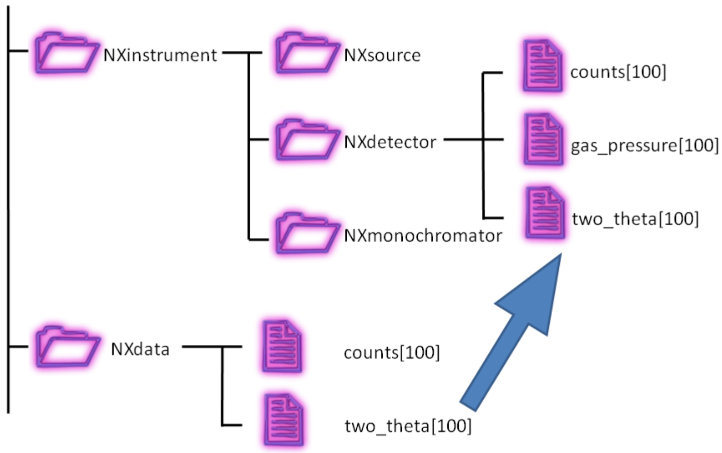


Fig. 1: Linking in a NeXus file

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[Page 21, 3] 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

[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* question: **I'm using links to place data in two places. Which one should be the data and which one is the link?**

```
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 satisifed 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:

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. 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

[6] https://github.com/nexusformat/code/raw/master/doc/api/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 **(superceded by** ***NXtransformations*****,**[Page 24, 7]**)**

`NXgeometry` and its subgroups `NXtranslation`, `NXorientation`, `NXshape` are used to store absolute 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:

base class	intent
NXnote	to store additional information
NXlog	information that is time-stamped
NXparameters	parameters for processing or analysis
NXcollection	to store *any* unvalidated content

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 properly 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). 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.

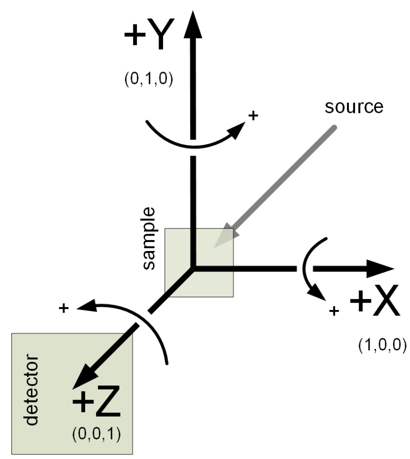


Fig. 2: NeXus coordinate system, as viewed from detector

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

Field Name	transformation_type	vector
polar_angle	rotation	0 1 0
azimuthal_angle	rotation	0 0 1
meridional_angle	rotation	1 0 0
distance	translation	0 0 1
height	translation	0 1 0
x_translation	translation	1 0 0
chi	rotation	0 0 1
phi	rotation	0 1 0

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).

```
OFF
#  cube.off
#  A cube

8 6 12
1.0   0.0   1.0
0.0   1.0   1.0
-1.0   0.0   1.0
0.0  -1.0   1.0
1.0   0.0  0.0
0.0   1.0  0.0
-1.0   0.0  0.0
0.0  -1.0  0.0
4  0 1 2 3
4  7 4 0 3
4  4 5 1 0
4  5 6 2 1
4  3 2 6 7
4  6 5 4 7
```

Following the initial line are the xyz coordinates of each vertex, followed by 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.

```
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,   -1.0,  1.0
    1.0,   0.0,   0.0
    0.0,   1.0,   0.0
    -1.0,  0.0,   0.0
    0.0,   -1.0,  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.

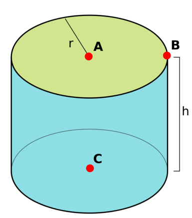


Fig. 3: Cylinder definition with three vertices

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:

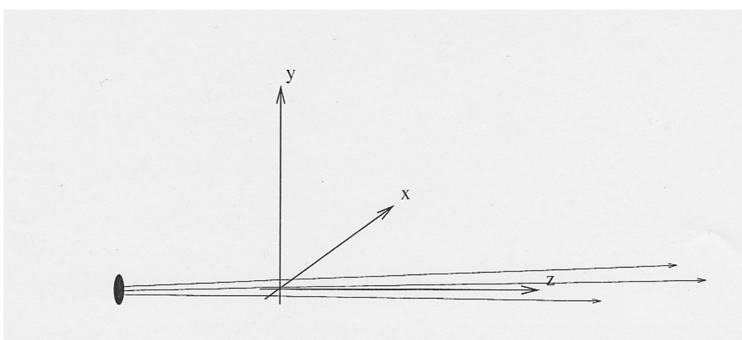


Fig. 4: The McStas Coordinate System

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*.

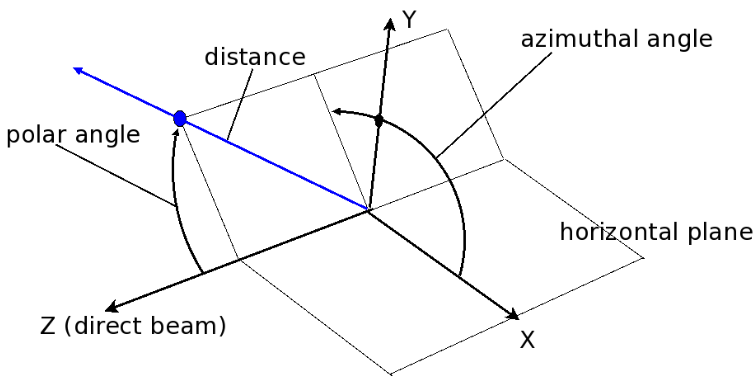


Fig. 5: NeXus Simple (Spherical Polar) Coordinate System

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

```
entry:NXentry
    @default = data
    instrument:NXinstrument
        source:NXsource
        ....
        detector:NXdetector
            data:NX_INT32[512,512]
    sample:NXsample
    control:NXmonitor
    data:NXdata
        @signal = data
        @axes = [".", "."]
        data --> /entry/instrument/detector/data
```

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

```
entry:NXentry
    @default = data
    reduction:NXprocess
        program_name = "pyDataProc2010"
        version = "1.0a"
        input:NXparameters
            filename = "sn2013287.nxs"
    sample:NXsample
```

(continues on next page)

(continued from previous page)

```
        data:NXdata
            @signal = data
            @axes = "."
            data
```

NeXus stores such data in a simplified `NXentry` structure. A processed data `NXentry` has at minimum a `NXsample`, a `NXdata` and a `NXprocess` group. Again the preferred name for the `NXsample` group is `sample`. In the case of processed data, the `NXdata` group holds the result of the processing together with the associated axis data. The `NXprocess` group holds the name and version of the program used for this processing step and further `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 `NXentry` can hold a `NXinstrument` group with further groups holding relevant information about the instrument. The preferred name is again `instrument`. Whereas for a raw data file, NeXus strives to capture as much data as possible, a `NXinstrument` group for processed data may contain a much-reduced subset.

`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 `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 `NXentry`.
- The `NXentry` group contains further `NXsubentry` groups, one for each method.
- Each *NXsubentry* group is constructed like a `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 `definition` field of the *NXsubentry* group
- Each `NXsubentry` group contains a `NXdata` group describing the default plottable data for that experimental method. To satisfy the NeXus requirement of finding the default plottable data from a `NXentry` group, the `NXdata` group from one of these `NXsubentry` groups (the fluoresence data) was linked.

See figure *NeXus Multi Method Hierarchy* for an example hierarchy. Note that `[,]` represents a 2D array.

NeXus Multi Method Hierarchy

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

(continues on next page)

(continued from previous page)

```
            definition = "NXsas"
            instrument:NXinstrument
                detector:NXdetector
                    data --> /entry/instrument/SASdet/data
            data:NXdata
                data --> /entry/instrument/SASdet/data
        Fluo:NXsubentry
            definition = "NXfluo"
            instrument:NXinstrument
                detector --> /entry/instrument/fluordet/data
                detector2 --> /entry/instrument/large_area/data
            data:NXdata
                @signal = detector
                @axes = [".", "."]
                detector --> /entry/instrument/fluordet/data
        data:NXdata --> /entry/Fluo/data
```

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

```
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

```
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 *hkl* scan

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

- A dataset at `NXentry/NXinstrument/NXdetector/data` of length `NP` containing the count data.
- A dataset at `NXentry/NXinstrument/NXdetector/polar_angle` of length `NP` containing the positions of the detector's polar_angle at the various steps of the scan.
- A dataset at `NXentry/NXsample/rotation_angle` of length `NP` containing the positions of `rotation_angle` at the various steps of the scan.
- A dataset at `NXentry/NXsample/chi` of length `NP` containing the positions of chi at the various steps of the scan.
- A dataset at `NXentry/NXsample/phi` of length `NP` containing the positions of phi at the various steps of the scan.
- A dataset at `NXentry/NXsample/h` of length `NP` containing the positions of the reciprocal coordinate h at the various steps of the scan.
- A dataset at `NXentry/NXsample/k` of length `NP` containing the positions of the reciprocal coordinate k at the various steps of the scan.
- A dataset at `NXentry/NXsample/l` of length `NP` containing the positions of the reciprocal coordinate l at the various steps of the scan.
- `NXdata` contains links to:
  - `NXentry/NXinstrument/NXdetector/data`
  - `NXentry/NXinstrument/NXdetector/polar_angle`
  - `NXentry/NXsample/rotation_angle`
  - `NXentry/NXsample/chi`
  - `NXentry/NXsample/phi`
  - `NXentry/NXsample/h`
  - `NXentry/NXsample/k`
  - `NXentry/NXsample/l`

  The `NXdata` also contains appropriate attributes as described in *Associating plottable data using attributes applied to the NXdata group*.
- All other fields have their normal dimensions.

NeXus Complex *hkl* Scan

```
entry:NXentry
    @default = data
    instrument:NXinstrument
        detector:NXdetector
            data[NP]
            polar_angle[NP]
            name
    sample:NXsample
```

(continues on next page)

(continued from previous page)

```
            name
            rotation_angle[NP]
            chi[NP]
            phi[NP]
            h[NP]
            k[NP]
            l[NP]
        control:NXmonitor
            data[NP]
        data:NXdata
            @signal = data
            @axes = "h"
            @h_indices = 0
            @k_indices = 0
            @l_indices = 0
            @chi_indices = 0
            @phi_indices = 0
            @polar_angle_indices = 0
            @rotation_angle_indices = 0
            data --> /entry/instrument/detector/data
            rotation_angle --> /entry/sample/rotation_angle
            chi --> /entry/sample/chi
            phi --> /entry/sample/phi
            polar_angle --> /entry/instrument/detector/polar_angle
            h --> /entry/sample/h
            k --> /entry/sample/k
            l --> /entry/sample/l
```

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 (`I0`) 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(I0/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

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

(continues on next page)

(continued from previous page)

```
                data:NX_NUMBER[NE,NT]
                energy --> /entry/monochromator/energy
                temperature --> /entry/sample/temperature
        sample:NXsample
            temperature:NX_NUMBER[NT]
        monochromator:NXmonochromator
            energy:NX_NUMBER[NE]
        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
        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
```

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, (orginal 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 *NXevent_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.

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
        sgu: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 *regular expression syntax*, as described below.

- For the class names[1] 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[3]
  - 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.[Page 41, 3]

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:

```
temperature
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:

```
unit_cell_abc
unit_cell_alphabetagamma
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.

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

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 `DATASET`, the following table lists the suffixes reserved by NeXus.

suffix	reference	meaning
`_end`	*NXtransformations*	end points of the motions that start with `DATASET`
`_errors`	*NXdata*	uncertainties (a.k.a., errors)
`_increment`	*NXtransformations*	intended average range through which the corresponding axis moves during the exposure of a frame
`_indices`	*NXdata*	Integer array that defines the indices of the signal field which need to be used in the `DATASET` in order to reference the corresponding axis value
`_mask`		Field containing a signal mask, where 0 means the pixel is not masked. If required, bit masks are defined in *NXdetector* `pixel_mask`.
`_set`	*target values*	Target value of `DATASET`
`_weights`		divide `DATASET` by these weights[4]

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:

```
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 `_errors` appended to it. These uncertainties fields have the same shape as the original data field.

An example, from NXdetector:

[4] If `DATASET_weights` exists and has the same shape as the field, you are supposed to divide `DATASET` by the weights.

```
data
data_errors
beam_center_x
beam_center_x_errors
```

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:

```
* 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 4 8 12 16
              1 5 9 13 17
              2 6 10 14 18
              3 7 11 15 19

  * "crazy reverse" row major (C) stride:
     stride[2] = { -5, -1 }
     offset[2] = { 4, 5 }
     for i:
        for j:
           result[i][j]:
              19 18 17 16 15
              14 13 12 11 10
              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)

(continued from previous 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

description	matching regular expression
integer	`NX_INT(8|16|32|64)`
floating-point	`NX_FLOAT(32|64)`
array	`(\\[0-9\\])?`
valid item name	`^[a-zA-Z0-9_]([a-zA-Z0-9_.]*[a-zA-Z0-9_])?$`
valid class name	`^NX[A-Za-z0-9_]*$`

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[6] 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*

[6] 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.

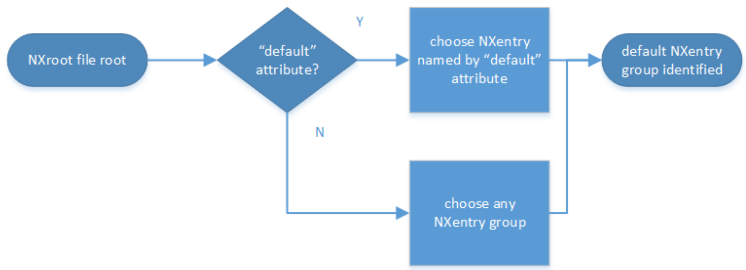


Fig. 6: Find plottable data: select the `NXentry` group

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.

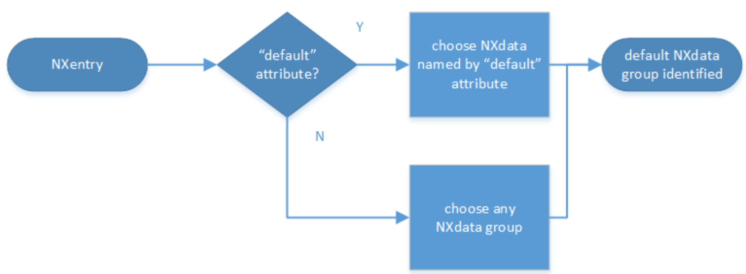


Fig. 7: Find plottable data: select the `NXdata` group

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.

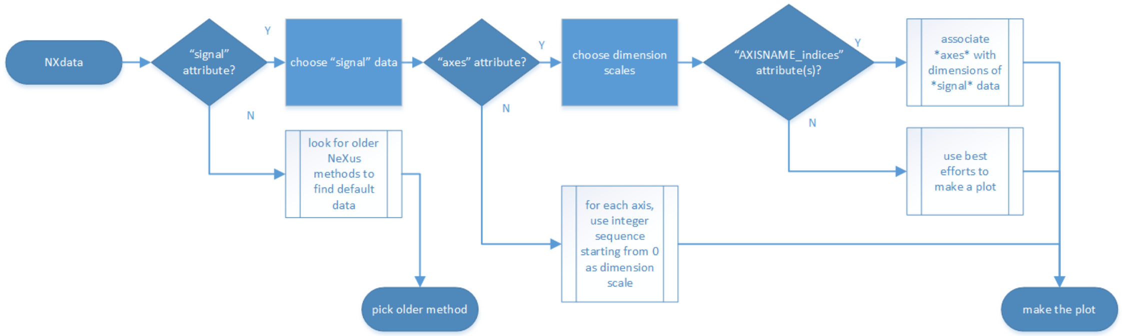


Fig. 8: Find plottable data: select the *signal* data

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 possibilty 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:

```
group = h5py.File(hdf5_file_name, "r")

while "default" in group.attrs:
    child_group_name = group.attrs["default"]
    group = group[child_group_name]

# assumes group.attrs["NX_class"] == "NXdata"
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.

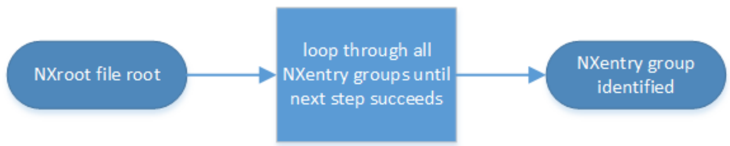


Fig. 9: Find plottable data: pick a `NXentry` group

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

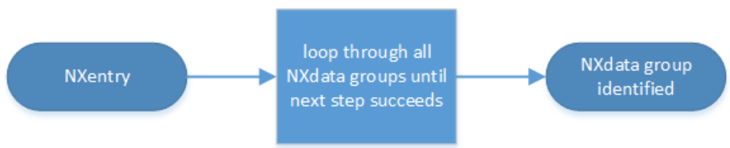


Fig. 10: Find plottable data: pick a `NXdata` group

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.

   This is the default plottable data.

   If there is no such `signal="1"` field, proceed to try an *older NeXus method* to find the default plottable data.

   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.

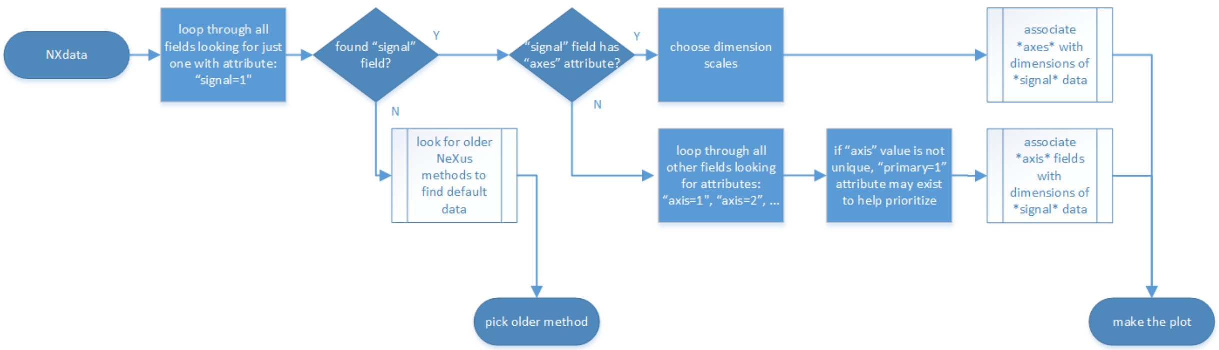


Fig. 11: Find plottable data: select the *signal* data

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`.

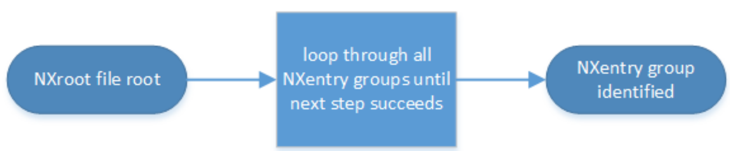


Fig. 12: Find plottable data: pick the first `NXentry` group

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

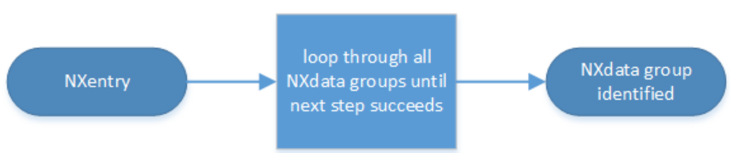


Fig. 13: Find plottable data: pick the first `NXdata` group

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.

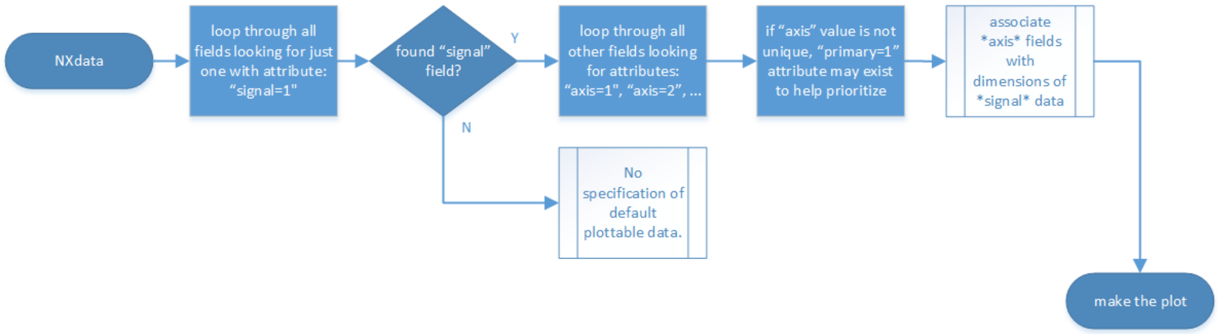


Fig. 14: Find plottable data: select the *signal* data

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.[7] 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[Page 54, 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:

```
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

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

denoting axes by integer number

```
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
  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)

```
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

```
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"
   }
}
```

h5dump of a NeXus link

```
# 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.

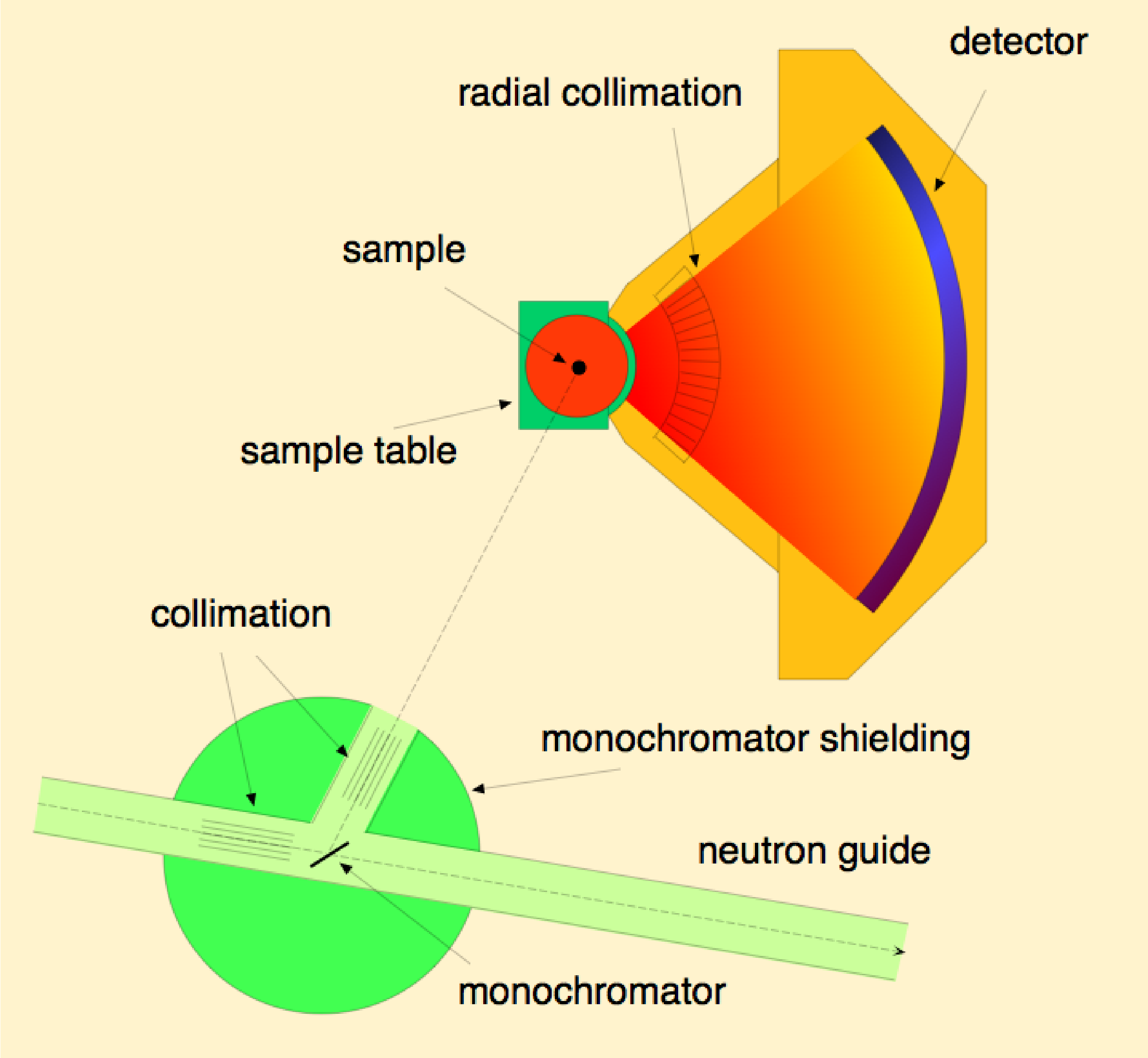


Fig. 15: The (fictional) WONI example powder diffractometer

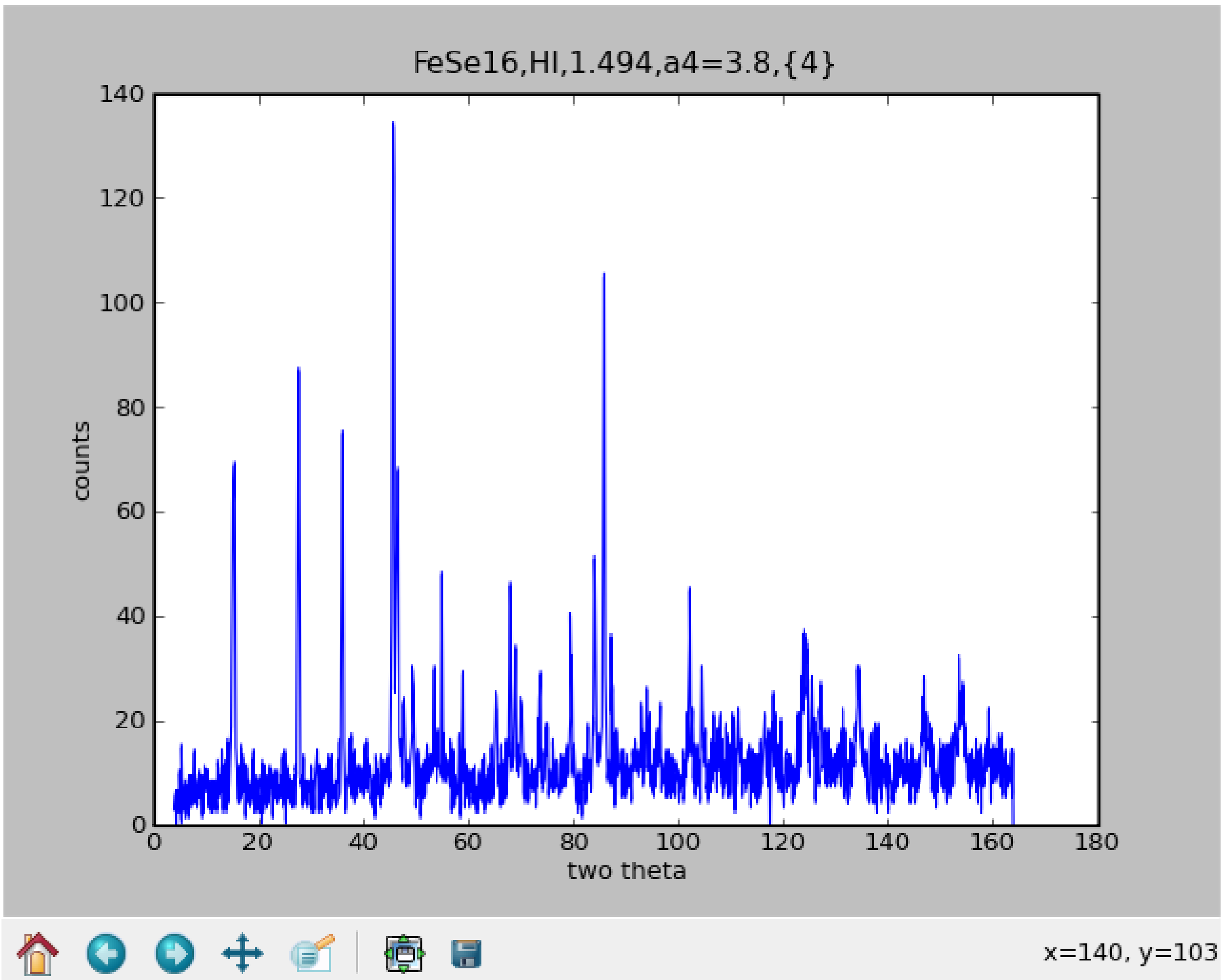


Fig. 16: Example Powder Diffraction Plot from (fictional) WONI at HYNES

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 components 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
```

(continues on next page)

(continued from previous page)

```
            reflection
            type
      NXmonitor
      NXsample
```

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 is if you are unsure. Perhaps you must call an international meeting with domain experts to haggle that out. When considering this, people tend to put in everything which might come up. This is not the way to go.

A key test question is: Is this data item necessary for common data analysis? Only these necessary data items belong in an application definition.

The purpose of an application definition is that an author of upstream software who consumes the file can expect certain data items to be there at well defined places. On the other hand if there is a development in your field which analyzes data in a novel way and requires more data to do it, then it is better to err towards the side of more data.

Now for the case of WONI, the standard data analysis is either Rietveld refinement or profile analysis. For both purposes, the kind of radiation used to probe the sample (for WONI, neutrons), the wavelength of the radiation, the monitor (which tells us how long we counted) used to normalize the data, the counts and the two theta angle of each detector element are all required. Usually, it is desirable to know what is being analyzed, so some metadata would be nice: a title, the sample name and the sample temperature. The data typically being plotted is two theta against counts, as shown in *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* (λ) of radiation incident on sample
- angle (2θ 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 then the first one. We need to map the data items which were collected in Step 1 into the NeXus hierarchy. A NeXus file hierarchy starts with an `NXentry` group. At this stage it is advisable to pull up the base class definition for `NXentry` and study it. The first thing you might notice is that `NXentry` contains a field named `title`. Reading the documentation, you quickly realize that this is a good place to store our title. So the first mapping has been found.

```
title = /NXentry/title
```

Note: In this example, the mapping descriptions just contain the path strings into the NeXus file hierarchy with the class names of the groups to use. As it turns out, this is the syntax used in NXDL link specifications. How convenient!

Another thing to notice in the `NXentry` base class is the existence of a group of class `NXsample`. This looks like a great place to store information about the sample. Studying the `NXsample` base class confirms this view and there are two new mappings:

```
sample name = /NXentry/NXsample/name
sample temperature = /NXentry/NXsample/temperature
```

Scanning the `NXentry` base class further reveals there can be a `NXmonitor` group at this level. Looking up the base class for `NXmonitor` reveals that this is the place to store our monitor information.

```
monitor = /NXentry/NXmonitor/data
```

For the other data items, there seem to be no solutions in `NXentry`. But each of these data items describe the instrument in more detail. NeXus stores instrument descriptions in the `/NXentry/NXinstrument` branch of the hierarchy. Thus, we continue by looking at the definition of the `NXinstrument` base class. In there we find further groups for all possible instrument components. Looking at the schematic of WONI (*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:

WONI data	NeXus path
title of measurement	/NXentry/title
sample *name*	/NXentry/NXsample/name
sample *temperature*	/NXentry/NXsample/temperature
monitor	/NXentry/NXmonitor/data
type of radiation *probe*	/NXentry/MXinstrument/NXsource/probe
wavelength of radiation incident on sample	/NXentry/MXinstrument/NXcrystal/wavelength
two theta of detector elements	/NXentry/NXinstrument/NXdetector/polar_angle
counts for each detector element	/NXentry/NXinstrument/NXdetector/data
two theta of detector elements	/NXentry/NXdata/polar_angle
counts for each detector element	/NXentry/NXdata/data

Looking at this 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 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)

(continued from previous page)

```
# 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 https://www.nexusformat.org/
-->
<definition name="NX__template__" extends="NXobject" type="group"
    category="application"
    xmlns="http://definition.nexusformat.org/nxdl/3.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://definition.nexusformat.org/nxdl/3.1 ../nxdl.xsd"
    >
    <doc>template for a NXDL application definition</doc>
</definition>
```

For example, copy and rename the file to `NXwoni.nxdl.xml`. Then, locate the XML root element `definition` and change the `name attribute` (the XML shorthand for this attribute is `/definition/@name`) to `NXwoni`. Change the `doc` as well.

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 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>
```

(continues on next page)

(continued from previous page)

```
        </symbol>
        <symbol name="nDet">
            <doc>Number of detectors</doc>
        </symbol>
    </symbols>
    <doc>
        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.
    </doc>
    <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 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
```

(continues on next page)

(continued from previous page)

```
                    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>
    </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 addition 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'lll have these positioners as arrays and assume that a default plot of *photodiode* vs. *ar*:

positioner arrays	detector arrays
`ar`, `ay`, `dy`	`I0`, `I00`, `time`, `Epoch`, `photodiode`

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

```
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

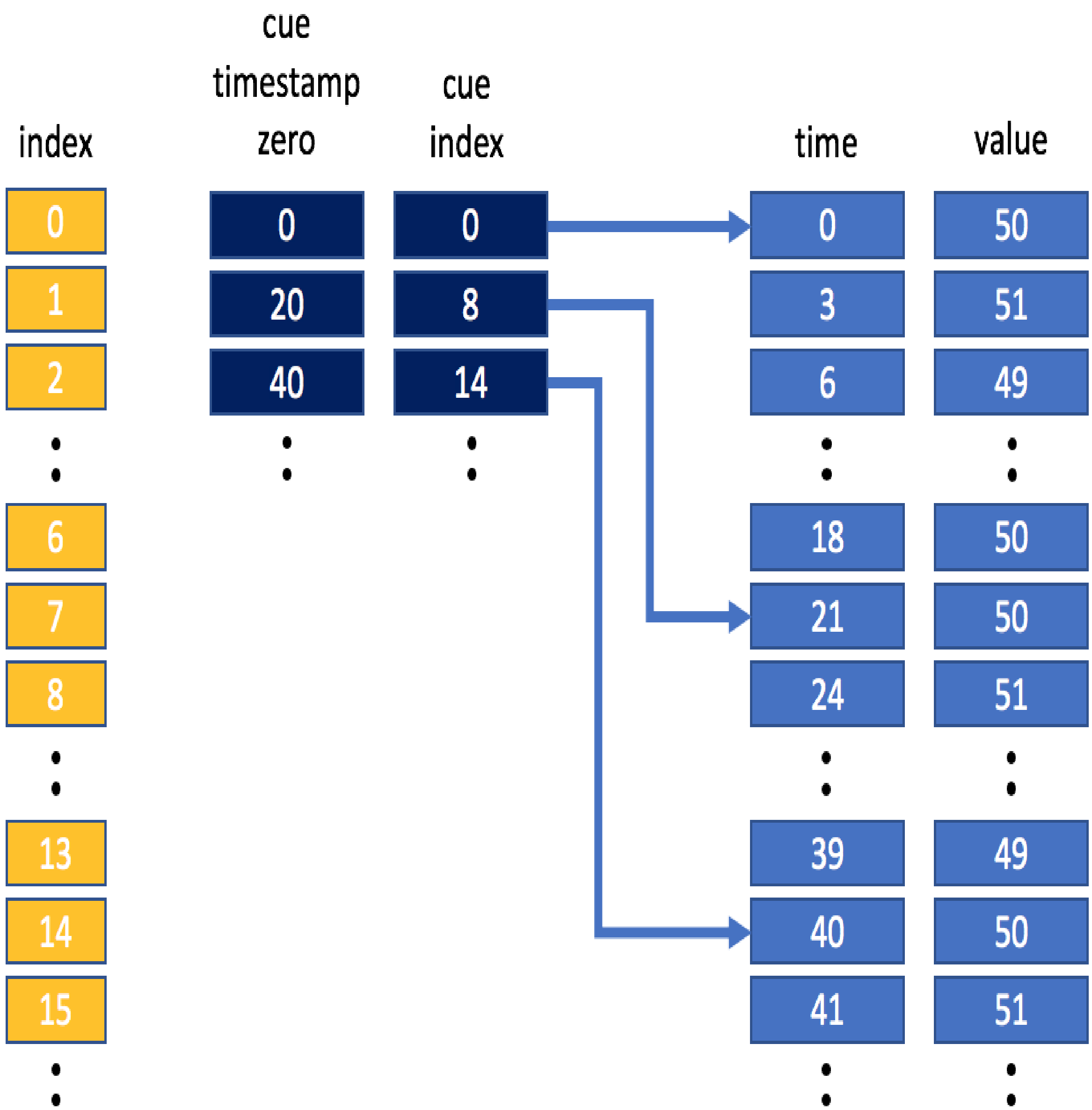
How should I store time-stamped data?

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.[1]

8. What is the purpose of `NXdata`?

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

`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*):

    ```
    <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).

CHAPTER

TWO

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.

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

(continues on next page)

(continued from previous 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];
```

(continues on next page)

(continued from previous page)

```
  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)

(continued from previous 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)

(continued from previous page)

```
  H5Fclose(fid);
}
```

Reading a simple NeXus file using native HDF5 commands in 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 = H5Dget_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);

```

(continues on next page)

(continued from previous page)

```
  /*
   * open and read the two_theta data set
   */
  dataid = H5Dopen(fid,"/scan/data/two_theta",H5P_DEFAULT);
  dataspace = H5Dget_space(dataid);
  rank = H5Sget_simple_extent_ndims(dataspace);
  dim = malloc(rank*sizeof(hsize_t));
  H5Sget_simple_extent_dims(dataspace, dim, NULL);
  two_theta = malloc(dim[0]*sizeof(float));
  memdataspace = H5Tcopy(H5T_NATIVE_FLOAT);
  H5Dread(dataid,memdataspace,H5S_ALL, H5S_ALL,H5P_DEFAULT, two_theta);
  H5Dclose(dataid);
  H5Sclose(dataspace);
  H5Tclose(memdataspace);



  H5Fclose(fid);

  for(i = 0; i < dim[0]; i++){
    printf("%8.2f %10d\n", two_theta[i], counts[i]);
  }

}
```

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`[1] and `h5py`[2] 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`[3] 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

Code examples

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

nexusformat

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

(continues on next page)

(continued from previous page)

```python
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)
```

h5py

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

from pathlib import Path
import datetime
import h5py  # HDF5 support
import numpy

print("Write a NeXus HDF5 file")
fileName = "simple_example_basic.nexus.hdf5"
timestamp = datetime.datetime.now().astimezone().isoformat()

# 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 h5py.File(fileName, "w") as f:
    # point to the default data to be plotted
    f.attrs["default"] = "entry"
    # give the HDF5 root some more attributes
    f.attrs["file_name"] = fileName
    f.attrs["file_time"] = timestamp
    f.attrs["instrument"] = "APS USAXS at 32ID-B"
    f.attrs["creator"] = "simple_example_basic_write.py"
    f.attrs["NeXus_version"] = "4.3.0"
    f.attrs["HDF5_Version"] = h5py.version.hdf5_version
    f.attrs["h5py_version"] = h5py.version.version

    # create the NXentry group
```

(continues on next page)

(continued from previous page)

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

    # create the NXentry group
    nxdata = nxentry.create_group("mr_scan")
    nxdata.attrs["NX_class"] = "NXdata"
    nxdata.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)
```

h5py

```
#!/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)

(continued from previous page)

```
23  17.9222 43710
24  17.9221 29315
25  17.9219 19782
26  17.9217 12992
27  17.9216 6622
28  17.9214 4198
29  17.9213 2248
30  17.9211 1321
```

downloads

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

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

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.

nexusformat

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

(continues on next page)

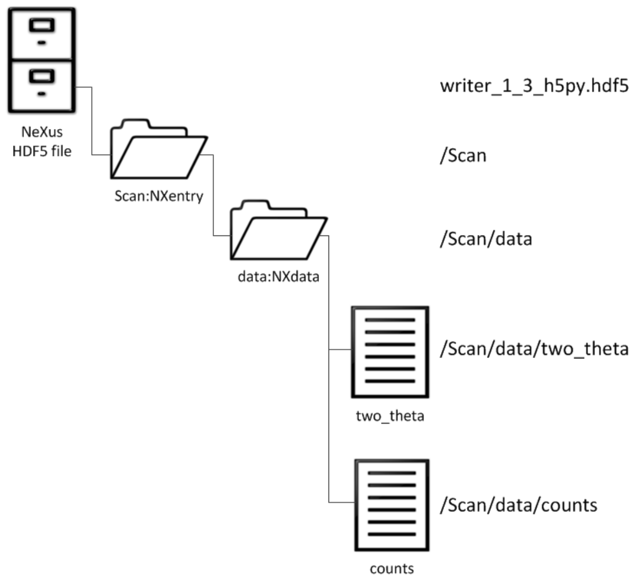


Fig. 1: Simple Example

(continued from previous page)

```
Writes the simplest NeXus HDF5 file using h5py

Uses method accepted at 2014NIAC
according to the example from Figure 1.3
in the Introduction chapter
"""

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:  # create the NeXus file
    f["Scan"] = NXentry()
    tth = NXfield(tthData, name="two_theta", units="degrees")
    counts = NXfield(countsData, name="counts", units="counts")
    f["Scan/data"] = NXdata(counts, tth)
```

h5py

```
#!/usr/bin/env python
"""
Writes the simplest NeXus HDF5 file using h5py

```

(continues on next page)

(continued from previous page)

```
Uses method accepted at 2014NIAC
according to the example from Figure 1.3
in the Introduction 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_write1.hdf5", "w") as f:  # create the HDF5 NeXus file
    # since this is a simple example, no attributes are used at this point

    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* tool[1] 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:

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

(continues on next page)

[1] *punx tree* : https://punx.readthedocs.io/en/latest/source_code/h5tree.html#how-to-use-h5tree

(continued from previous page)

```
    @NX_class = "NXdata"
    @axes = "two_theta"
    @signal = "counts"
    @two_theta_indices = [0]
    counts:NX_INT32[31] = [1037, 1318, 1704, '...', 1321]
      @units = "counts"
    two_theta:NX_FLOAT64[31] = [17.92608, 17.92591, 17.92575, '...', 17.92108]
      @units = "degrees"
```

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.

file	description
`../simple_example.dat`	2-column ASCII data used in this section
`simple_example_write1.py`	h5py code to write example *simple_example_write1*
`nexusformat/simple_example_write1.py`	nexusformat code to write example *simple_example_write1*
`simple_example_write1.hdf5`	NeXus file written by this code
`simple_example_write1_h5dump.txt`	*h5dump* analysis of the NeXus file
`simple_example_write1_structure.txt`	*punx tree* analysis of the NeXus file

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.

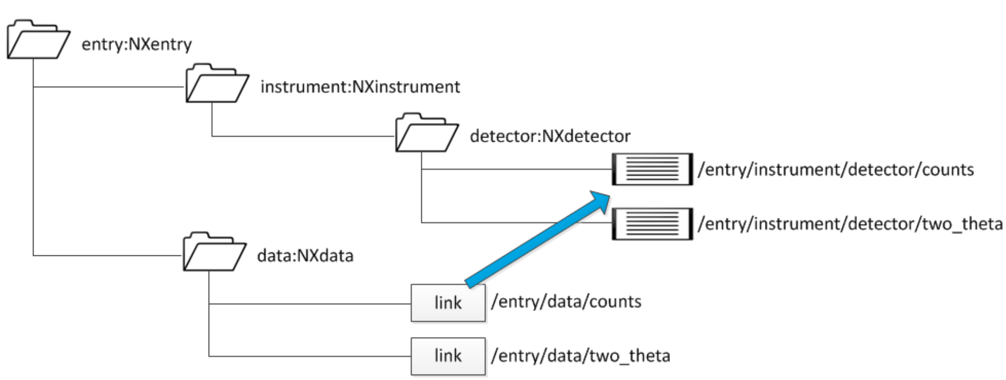


Fig. 2: h5py example showing linking in a NeXus file

The Python code to build an HDF5 data file with that structure (using numerical data from the previous example) is shown below.

nexusformat

```
#!/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()
```

h5py

```
#!/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 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[]
```

(continues on next page)

(continued from previous page)

```

with h5py.File("simple_example_write2.hdf5", "w") as f:  # create the HDF5 NeXus 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 description.

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.

```
@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.

file	description
`../simple_example.dat`	2-column ASCII data used in this section
`simple_example_write2.py`	h5py code to write example *simple_example_write2*
`nexusformat/simple_example_write2.py`	nexusformat code to write example *simple_example_write2*
`simple_example_write2.hdf5`	NeXus file written by this code
`simple_example_write2_h5dump.txt`	*h5dump* analysis of the NeXus file
`simple_example_write2_structure.txt`	*punx tree* analysis of the NeXus file

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.

```
f[local_addr] = h5py.ExternalLink(external_file_name, external_addr)
```

where f is an open h5py.File() object in which we will create the new link, local_addr is an HDF5 path address, external_file_name is the name (relative or absolute) of an existing HDF5 file, and external_addr is the HDF5 path address of the existing data in the external_file_name to be linked.

file: external_angles.hdf5

Take for example, the structure of external_angles.hdf5, a simple HDF5 data file that contains just the 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:

```
angles:float64[31] = [17.926079999999999, '...', 17.92108]
  @units = degrees
```

file: external_counts.hdf5

The data in the file external_angles.hdf5 might be referenced from another HDF5 file (such as 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, '...', 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.`

To be a valid NeXus file, it must contain a *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 = 0
```

Here is (the basic structure of) external_master.hdf5, an example:

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

(continues on next page)

[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

(continued from previous page)

```
  data:NXdata
    @signal = counts
    @axes = "two_theta"
     @two_theta = 0
    counts --> file="external_counts.hdf5", path="/entry/instrument/detector/counts"
    two_theta --> file="external_angles.hdf5", path="/angles"
```

source code: external_example_write.py

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

nexusformat

```python
#!/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

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)

(continued from previous 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")
```

(continues on next page)

(continued from previous page)

```python
    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"
    ds = nxdetector.create_dataset("counts", data=countsData)
    ds.attrs["units"] = "counts"
    # link the "two_theta" data stored in separate file
    local_addr = nxdetector.name + "/two_theta"
    f[local_addr] = h5py.ExternalLink(FILE_HDF5_ANGLES, "/angles")

# create a master NeXus HDF5 file
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"

    # link in the signal data
    local_addr = "/entry/data/counts"
    external_addr = "/entry/instrument/detector/counts"
    f[local_addr] = h5py.ExternalLink(FILE_HDF5_COUNTS, external_addr)
    nxdata.attrs["signal"] = "counts"

    # link in the axes data
    local_addr = "/entry/data/two_theta"
    f[local_addr] = h5py.ExternalLink(FILE_HDF5_ANGLES, "/angles")
    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.

file	description
`external_angles_h5dump.txt`	*h5dump* analysis of *external_angles.hdf5*
`external_angles.hdf5`	HDF5 file written by *external_example_write*
`external_angles_structure.txt`	*punx tree* analysis of *external_angles.hdf5*
`external_counts_h5dump.txt`	*h5dump* analysis of *external_counts.hdf5*
`external_counts.hdf5`	HDF5 file written by *external_example_write*
`external_counts_structure.txt`	*punx tree* analysis of *external_counts.hdf5*
`external_example_write.py`	h5py code to write external linking examples
`nexusformat/external_example_write.py`	nexusformat code to write external linking examples
`external_master_h5dump.txt`	*h5dump* analysis of *external_master.hdf5*
`external_master.hdf5`	NeXus file written by *external_example_write*
`external_master_structure.txt`	*punx tree* analysis of *external_master.hdf5*

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

nexusformat

```
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

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}")
    print(f"{axes.name} {signal.name}")
    for x, y in zip(axes, signal):
        print(x, y)
```

Output from `reader_attributes_trail.py` is shown next.

Output from `reader_attributes_trail.py`

```
file: simple_example_basic.nexus.hdf5
signal: /entry/mr_scan/I00
axes: /entry/mr_scan/mr
/entry/mr_scan/mr /entry/mr_scan/I00
17.92608 1037
17.92591 1318
17.92575 1704
17.92558 2857
17.92541 4516
17.92525 9998
17.92508 23819
17.92491 31662
17.92475 40458
17.92458 49087
17.92441 56514
17.92425 63499
```

(continues on next page)

(continued from previous page)

```
17.92408 66802
17.92391 66863
17.92375 66599
17.92358 66206
17.92341 65747
17.92325 65250
17.92308 64129
17.92291 63044
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.

file	description
`reader_attributes_trail.py`	h5py code to read NeXus HDF5 file and find plottable data
`nexusformat/reader_attributes_trail.py`	nexusformat code to read NeXus HDF5 file and find plottable data

- 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

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

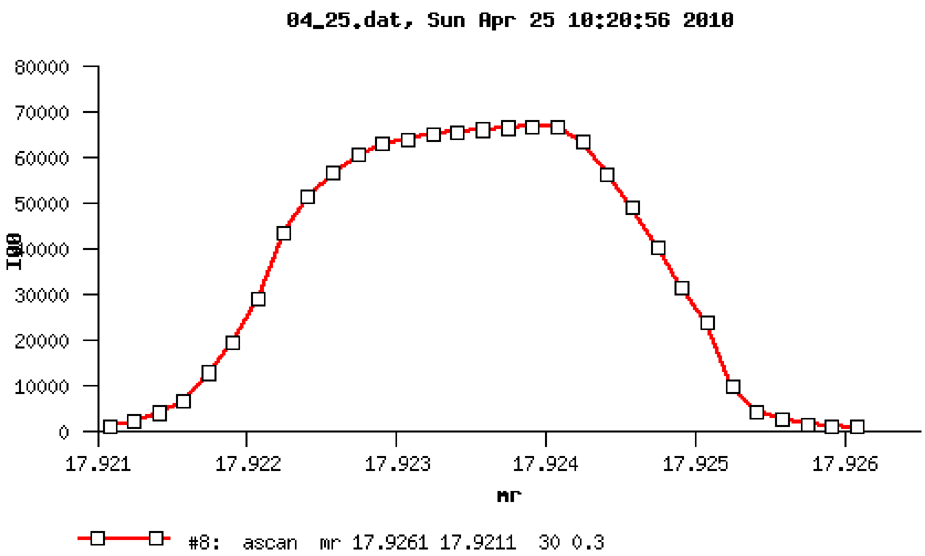


Fig. 4: plot of the simple example data

Simple example values

```
17.92608    1037
17.92591    1318
17.92575    1704
17.92558    2857
17.92541    4516
17.92525    9998
17.92508    23819
17.92491    31662
17.92475    40458
17.92458    49087
17.92441    56514
17.92425    63499
17.92408    66802
17.92391    66863
17.92375    66599
17.92358    66206
17.92341    65747
17.92325    65250
17.92308    64129
17.92291    63044
17.92275    60796
```

(continues on next page)

(continued from previous page)

```
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
```

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*.

```
17.92608    1037
17.92591    1318
17.92575    1704
17.92558    2857
17.92541    4516
17.92525    9998
17.92508    23819
17.92491    31662
17.92475    40458
17.92458    49087
17.92441    56514
17.92425    63499
17.92408    66802
17.92391    66863
17.92375    66599
17.92358    66206
17.92341    65747
17.92325    65250
17.92308    64129
17.92291    63044
17.92275    60796
17.92258    56795
17.92241    51550
17.92225    43710
17.92208    29315
```

(continues on next page)

(continued from previous page)

```
17.92191    19782
17.92175    12992
17.92158    6622
17.92141    4198
17.92125    2248
17.92108    1321
```

writing data

basic_writer.m: Write a NeXus HDF5 file using Matlab

```
% Writes a NeXus HDF5 file using matlab

disp 'Write a NeXus HDF5 file'
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,'/entry','default','mr_scan');
h5writeatt(filename,'/entry/mr_scan','signal','I00');
h5writeatt(filename,'/entry/mr_scan','axes','mr_scan');
h5writeatt(filename,'/entry/mr_scan','mr_scan_indices', int32(0));

% add NeXus metadata
```

(continues on next page)

(continued from previous page)

```
h5writeatt(filename,'/','file_name',filename);
h5writeatt(filename,'/','file_time',timestamp);
h5writeatt(filename,'/','instrument','APS USAXS at 32ID-B');
h5writeatt(filename,'/','creator','basic_writer.m');
h5writeatt(filename,'/','NeXus_version','4.3.0');
h5writeatt(filename,'/','HDF5_Version','1.6'); % no 1.8 features used in this example
h5writeatt(filename,'/entry','NX_class','NXentry');
h5writeatt(filename,'/entry/mr_scan','NX_class','NXdata');


h5disp(filename);
```

reading data

basic_reader.m: Read a NeXus HDF5 file using Matlab

```
% Reads NeXus HDF5 file and print the contents

filename = 'prj_test.nexus.hdf5';
root = h5info(filename,'/');
attrs = root.Attributes;
for i = 1:length(attrs)
    fprintf('%s: %s\n', attrs(i).Name, attrs(i).Value);
end
mr = h5read(filename,'/entry/mr_scan/mr');
i00 = h5read(filename, '/entry/mr_scan/I00');
fprintf('#\t%s\t%s\n','mr','I00');
for i = 1:length(mr)
    fprintf('%d\t%g\t%d\n', i, mr(i), i00(i));
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

filename = 'writer_2_1.hdf5';

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

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

% store x
```

(continues on next page)

(continued from previous page)

```
h5create(filename,'/entry/instrument/detector/two_theta',[length(two_theta)]);
h5write(filename,'/entry/instrument/detector/two_theta',two_theta);
h5writeatt(filename,'/entry/instrument/detector/two_theta','units','degrees');

% store y
h5create(filename,'/entry/instrument/detector/counts',[length(counts)],'DataType','int32
↪');
h5write(filename,'/entry/instrument/detector/counts',counts);
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');
h5link(filename,'/entry/instrument/detector/counts','/entry/data/counts');

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

% add NeXus metadata
h5writeatt(filename,'/','file_name',filename);
h5writeatt(filename,'/','file_time',timestamp);
h5writeatt(filename,'/','instrument','APS USAXS at 32ID-B');
h5writeatt(filename,'/','creator','writer_2_1.m');
h5writeatt(filename,'/','NeXus_version','4.3.0');
h5writeatt(filename,'/','HDF5_Version','1.6'); % no 1.8 features used in this example
h5writeatt(filename,'/entry','NX_class','NXentry');
h5writeatt(filename,'/entry/instrument','NX_class','NXinstrument');
h5writeatt(filename,'/entry/instrument/detector','NX_class','NXdetector');
h5writeatt(filename,'/entry/data','NX_class','NXdata');

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

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');
%
```

(continues on next page)

(continued from previous page)

```
%   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

file	description
`input.dat`	two-column text data file, also used in other examples
`basic_writer.m`	writes a NeXus HDF5 file using `input.dat`
`basic_reader.m`	reads the NeXus HDF5 file written by `basic_writer.m`
`h5link.m`	support module for creating NeXus-style HDF5 hard links
`writer_2_1.m`	like `basic_writer.m` but stores data in `/entry/instrument/detector` and then links to **NXdata** group

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 `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 C 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*.

```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);
          NXclosedata (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);
          NXclosedata (file_id);
/* Output data */
```

(continues on next page)

(continued from previous page)

```
          dims[0] = n_t;
          dims[1] = n_p;
          NXmakedata (file_id, "counts", NX_INT32, 2, dims);
          NXopendata (file_id, "counts");
            NXputdata (file_id, counts);
            i = 1;
            NXputattr (file_id, "signal", &i, 1, NX_INT32);
            NXputattr (file_id, "axes",  "polar_angle:time_of_flight", 26, NX_CHAR);
          NXclosedata (file_id);
/* Close NXentry and NXdata groups and close file */
        NXclosegroup (file_id);
      NXclosegroup (file_id);
    NXclose (&file_id);
    return;
}
```

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.

```
      program WRITEDATA

      include 'NAPIF.INC'
      integer*4 status, file_id(NXHANDLESIZE), counts(1000,50), n_p, n_t, dims(2)
      real*4 t(1000), phi(50)

!Read in data using local routines
      call getdata (n_t, t, n_p, phi, counts)
!Open output file
      status = NXopen ('NXFILE.NXS', NXACC_CREATE, file_id)
        status = NXputcharattr
     +         (file_id, 'user', 'Joe Bloggs', 10, NX_CHAR)
!Open top-level NXentry group
        status = NXmakegroup (file_id, 'Entry1', 'NXentry')
        status = NXopengroup (file_id, 'Entry1', 'NXentry')
!Open NXdata group within NXentry group
          status = NXmakegroup (file_id, 'Data1', 'NXdata')
          status = NXopengroup (file_id, 'Data1', 'NXdata')
!Output time channels
            status = NXmakedata
```

(continues on next page)

(continued from previous page)

```
     +         (file_id, 'time_of_flight', NX_FLOAT32, 1, n_t)
            status = NXopendata (file_id, 'time_of_flight')
              status = NXputdata (file_id, t)
              status = NXputcharattr
     +         (file_id, 'units', 'microseconds', 12, NX_CHAR)
            status = NXclosedata (file_id)
!Output detector angles
            status = NXmakedata (file_id, 'polar_angle', NX_FLOAT32, 1, n_p)
            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
            dims(1) = n_t
            dims(2) = n_p
            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
```

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*.

```
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
```

(continues on next page)

(continued from previous page)

```
!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")
```

(continues on next page)

(continued from previous page)

```
nf.opengroup("entry","NXentry")

nf.makegroup("data","NXdata")
nf.opengroup("data","NXdata")
nf.putattr("signal","test")

nf.makedata("test",'int32',[2,3,4])
nf.opendata("test")
nf.putdata(a)
nf.closedata()

nf.closegroup() # NXdata
nf.closegroup() # NXentry

nf.close()

exit
```

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 `mr` 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*

```
simple3D.h5:NeXus data file
  @NeXus_version = 4.1.0
  @file_name = simple3D.h5
```

(continues on next page)

[1] *NeXpy*: http://nexpy.github.io/nexpy/

[1] **silx view** : http://www.silx.org/doc/silx/latest/applications/view.html

[2] **silx** : http://www.silx.org/doc/silx/latest/

[1] **punx tree** : https://punx.readthedocs.io/en/latest/source_code/h5tree.html#how-to-use-h5tree

[2] **punx** : https://punx.readthedocs.io/

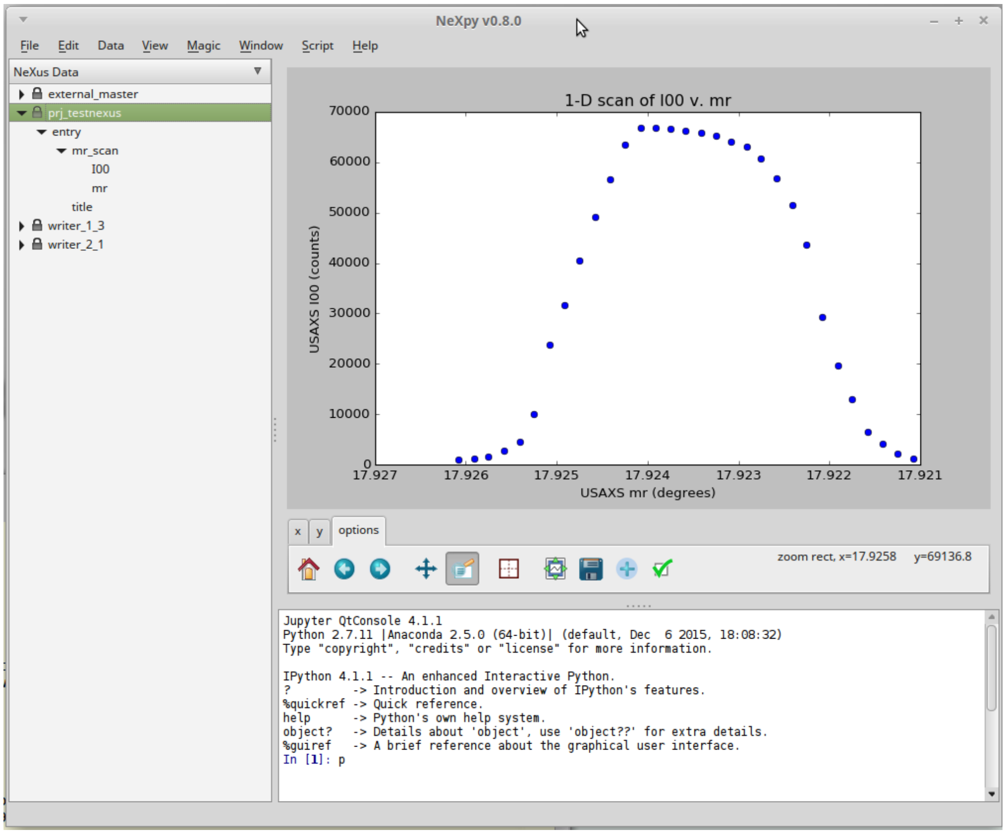


Fig. 5: plot the simple example using NeXpy

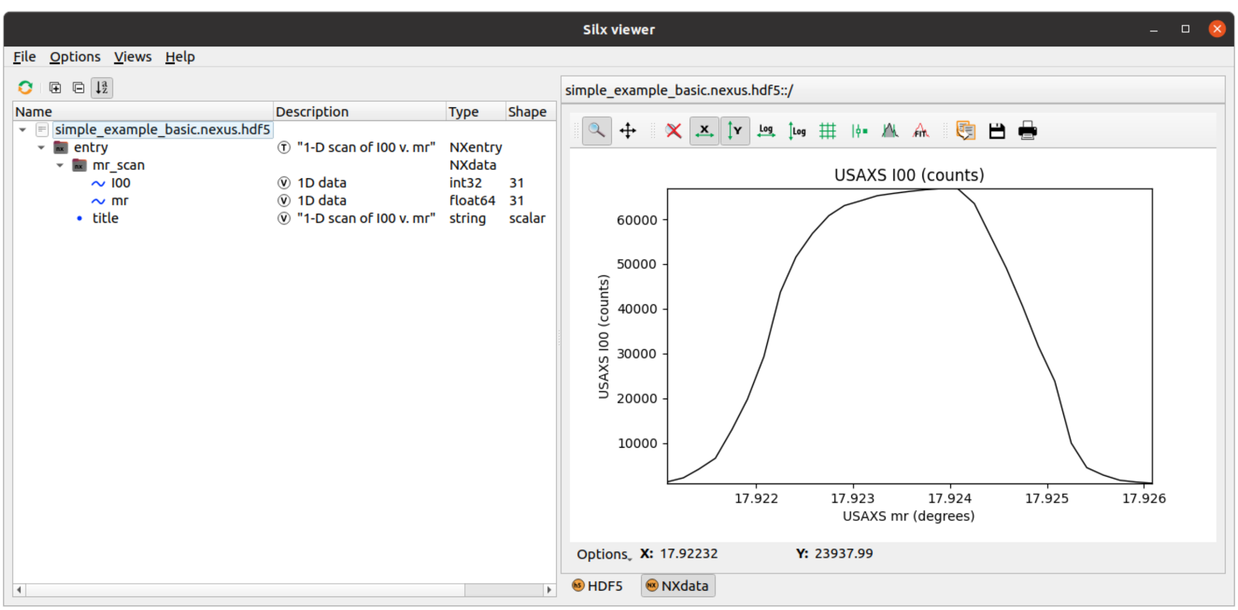


Fig. 6: plot the simple example using silx

(continued from previous page)

```
  @HDF5_Version = 1.6.6
  @file_time = 2011-11-18 17:26:27+0100
  entry:NXentry
    @NX_class = NXentry
    data:NXdata
      @NX_class = NXdata
      test:NX_INT32[2,3,4] = __array
        @signal = 1
        __array = [
            [
              [0, 1, 2, 3]
              [4, 5, 6, 7]
              [8, 9, 10, 11]
            ]
            [
              [12, 13, 14, 15]
              [16, 17, 18, 19]
              [20, 21, 22, 23]
            ]
          ]
```

2.2.4 View a NeXus HDF5 file with *h5dump*

The h5dump tool[1] provided as part of the HDF5 tool kit[2] can be used to print the content of an HDF5 file. As an example we show the result of the command h5dump simple3D.h5 on the result of *HDF5 in Python with NAPI*

```
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
```

(continues on next page)

[1] **h5dump** : https://support.hdfgroup.org/HDF5/doc/RM/Tools.html#Tools-Dump
[2] **HDF5 tools** : https://support.hdfgroup.org/products/hdf5_tools/

(continued from previous page)

```
      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"
```

(continues on next page)

(continued from previous page)

```
            }
         }
         DATASET "test" {
            DATATYPE  H5T_STD_I32LE
            DATASPACE  SIMPLE { ( 2, 3, 4 ) / ( 2, 3, 4 ) }
            DATA {
            (0,0,0): 0, 1, 2, 3,
            (0,1,0): 4, 5, 6, 7,
            (0,2,0): 8, 9, 10, 11,
            (1,0,0): 12, 13, 14, 15,
            (1,1,0): 16, 17, 18, 19,
            (1,2,0): 20, 21, 22, 23
            }
            ATTRIBUTE "signal" {
               DATATYPE  H5T_STD_I32LE
               DATASPACE  SCALAR
               DATA {
               (0): 1
               }
            }
         }
      }
   }
}
}
```

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.[1] For this example, we will start with a conversion of that original data file into *HDF5* format.

format	file name
HDF4	lrcs3701.nxs
HDF5	lrcs3701.nx5

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 `h5dump`

Here, the output of the command:

```
h5dump -H lrcs3701.nx5
```

has been edited to only show the first *NXdata* group (`/Histogram1/data`):

LRMECS `lrcs3701` data: `h5dump` output

```
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 ) }
   }
}
}
```

Visualize Using *HDFview*

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

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

Note: In this image, dark represents low intensity while white represents high intensity.

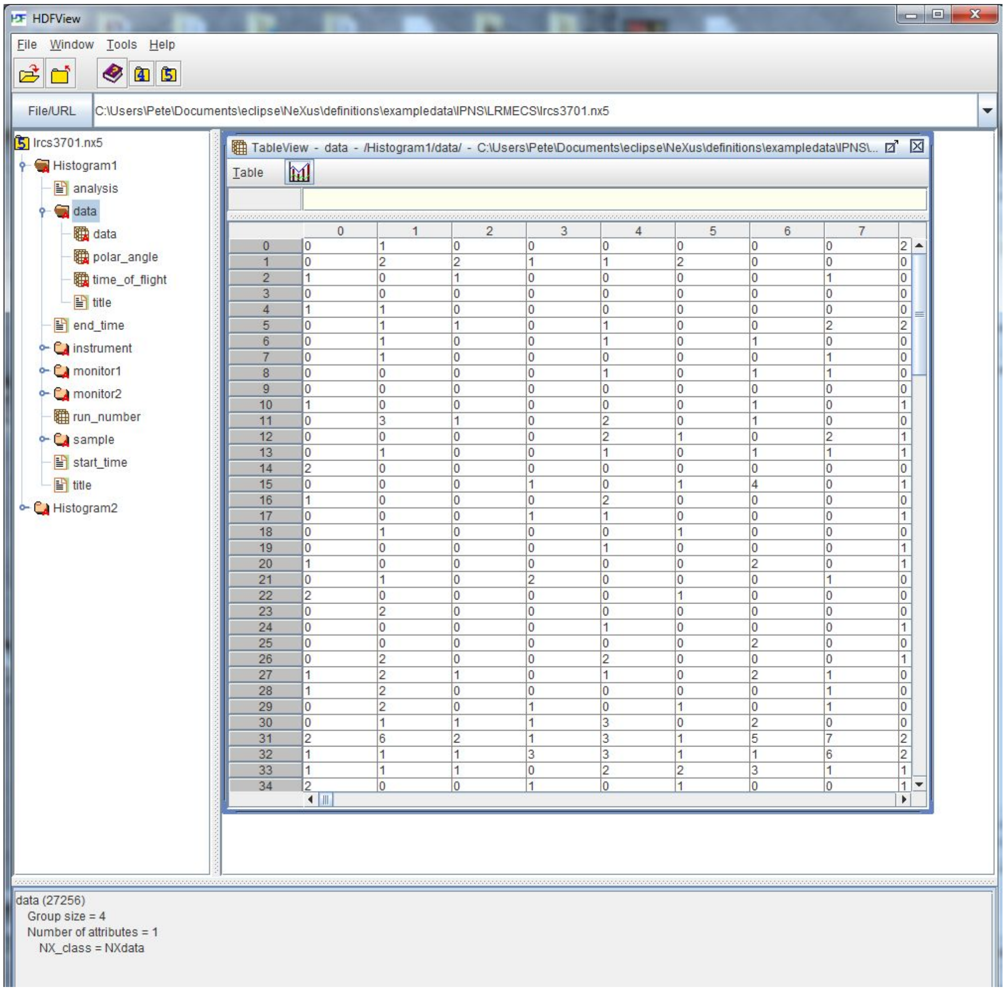


Fig. 7: LRMECS `lrcs3701` data: *HDFview*

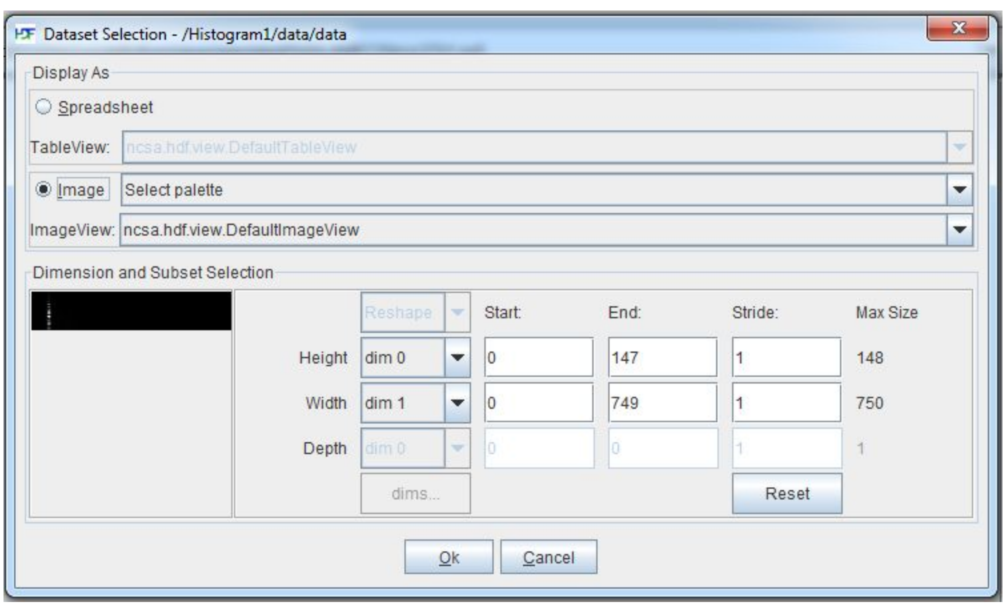


Fig. 8: LRMECS `lrcs3701` data: *HDFview Open As* dialog

LRMECS `lrcs3701` data: image

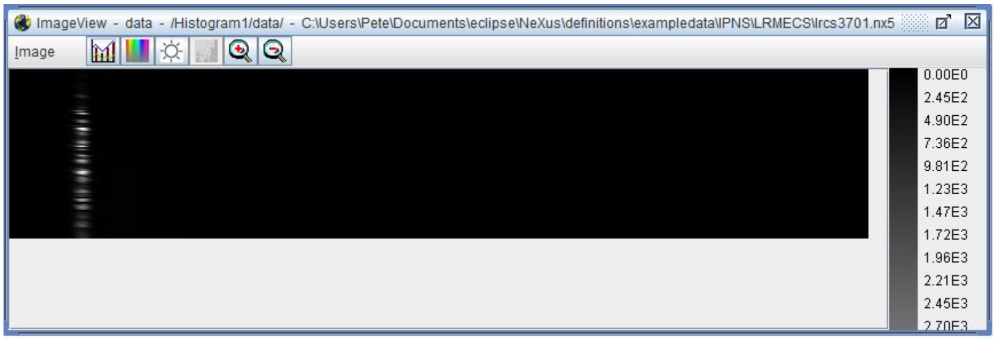


Fig. 9: LRMECS `lrcs3701` data: *HDFview* Image

Visualize Using *IgorPro*

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.

LRMECS `lrcs3701` data: 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.

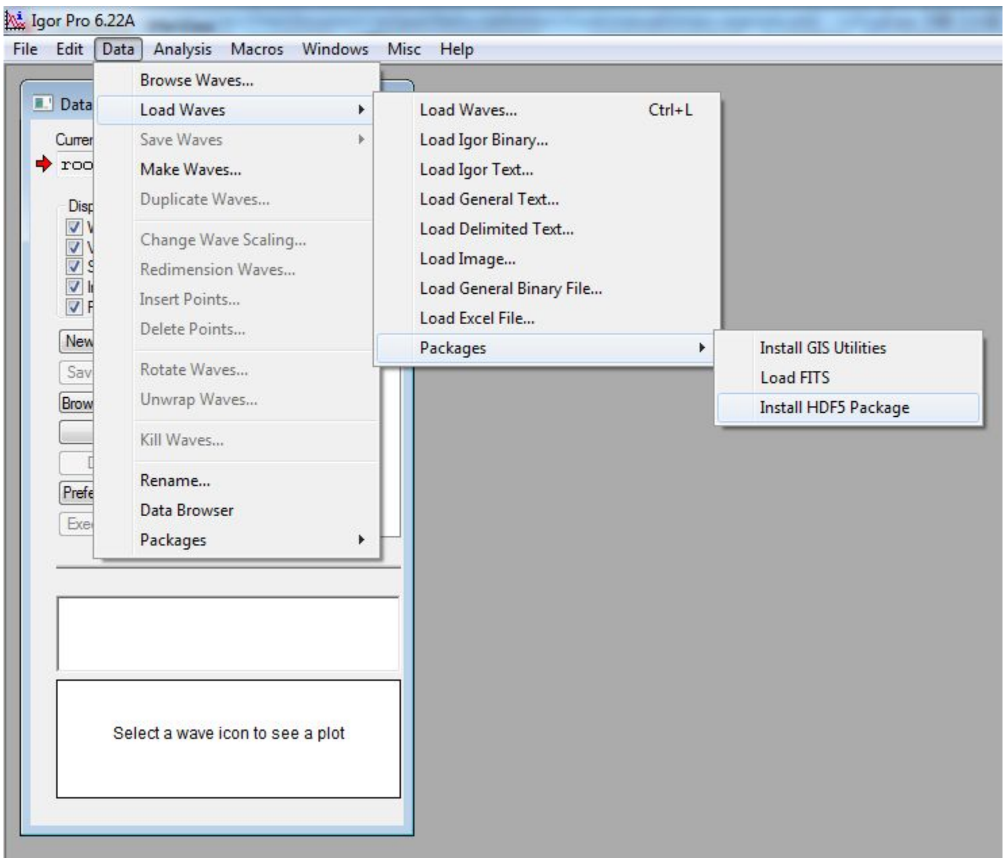


Fig. 10: LRMECS `lrcs3701` data: *IgorPro* install HDF5 Browser

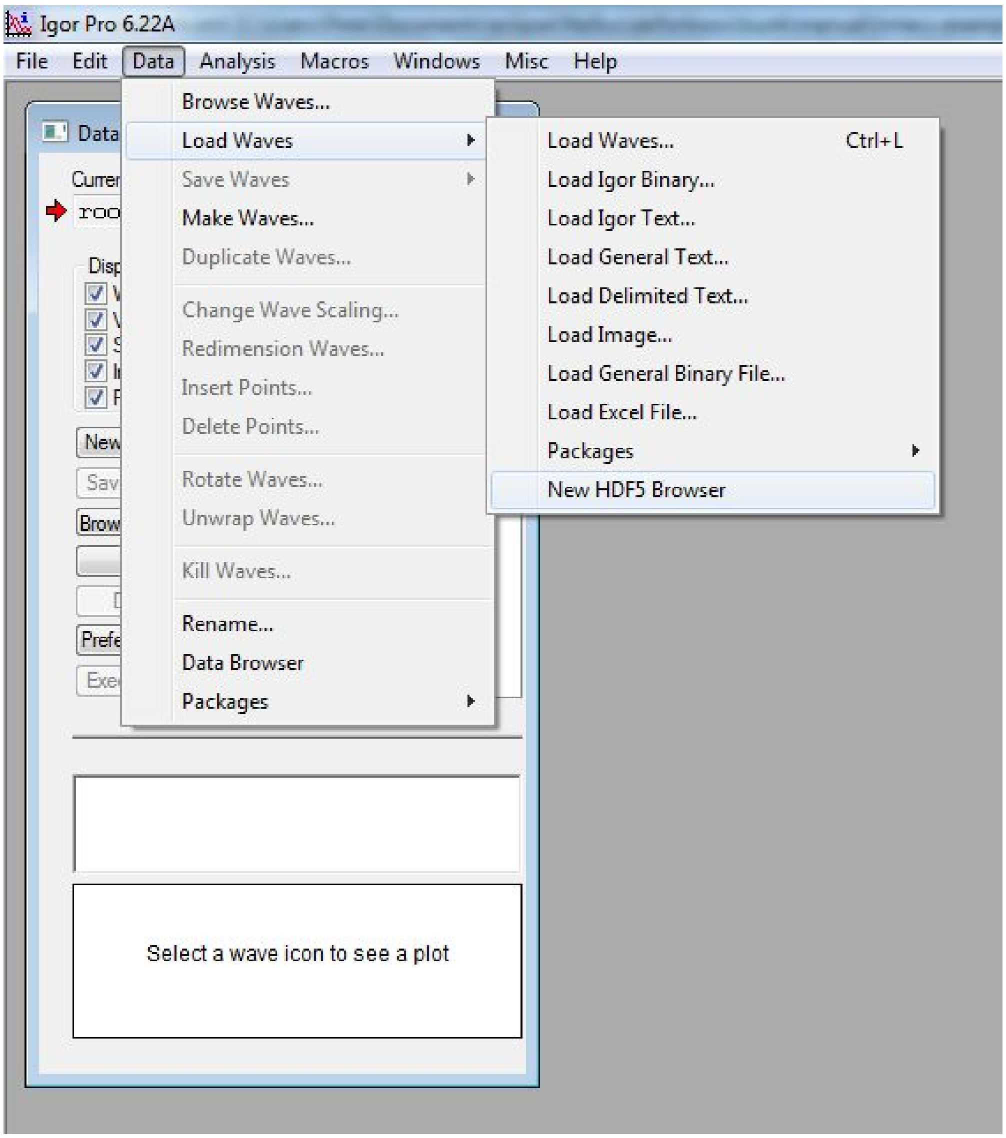


Fig. 11: LRMECS `lrcs3701` data: *IgorPro HDFBrowser* dialog

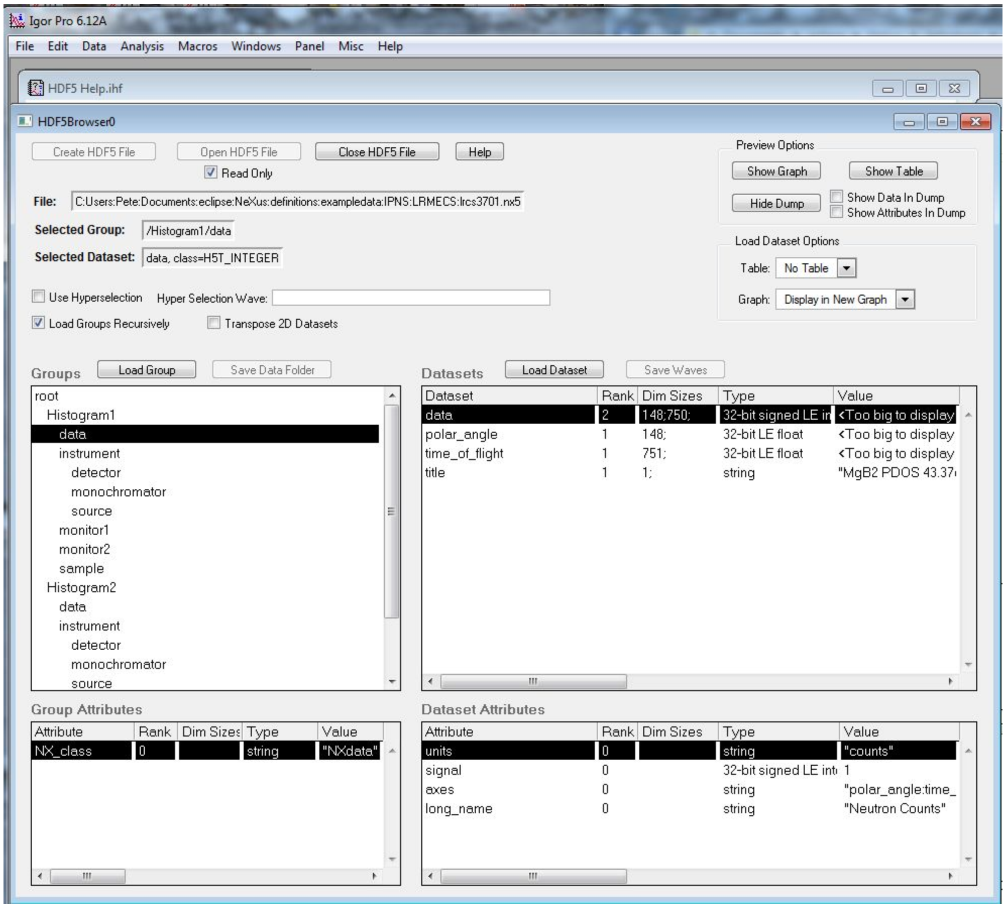


Fig. 12: LRMECS `lrcs3701` data: *IgorPro HDFBrowser* dialog

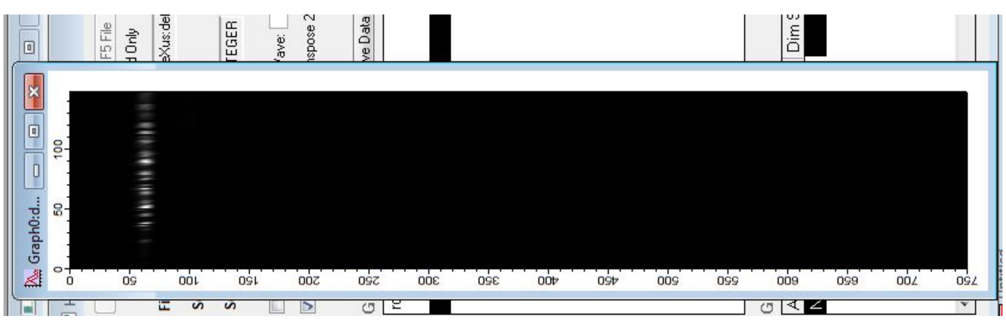


Fig. 13: LRMECS `lrcs3701` data: *IgorPro* Image

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:

file	description
`attributes.xml`	what information to know about from EPICS and other sources
`layout.xml`	where to write that information in the HDF5 file

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"
    >
    <Attribute name="AcquireTime"
               type="EPICS_PV"
               source="13SIM1:cam1:AcquireTime"
               dbrtype="DBR_NATIVE"
               description="Camera acquire time"/>
    <Attribute name="ImageCounter"
```

(continues on next page)

[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

(continued from previous page)

```
                type="PARAM"
                source="ARRAY_COUNTER"
                datatype="INT"
                description="Image counter"/>
    <Attribute name="calc1_val"
                type="EPICS_PV"
                source="prj:userCalc1.VAL"
                datatype="DBR_NATIVE"
                description="some calculation result"/>
    <Attribute name="calc2_val"
                type="EPICS_PV"
                source="prj:userCalc2.VAL"
                datatype="DBR_NATIVE"
                description="another calculation result"/>
    <Attribute name="MaxSizeX"
                type="PARAM"
                source="MAX_SIZE_X"
                datatype="INT"
                description="Detector X size"/>
    <Attribute name="MaxSizeY"
                type="PARAM"
                source="MAX_SIZE_Y"
                datatype="INT"
                description="Detector Y size"/>
    <Attribute name="CameraModel"
                type="PARAM"
                source="MODEL"
                datatype="STRING"
                description="Camera model"/>
    <Attribute name="CameraManufacturer"
                type="PARAM"
                source="MANUFACTURER"
                datatype="STRING"
                description="Camera manufacturer"/>
</Attributes>
```

If you want to add additional EPICS process variables (PVs) to be written in the HDF5 file, create additional `<Attribute />` elements (such as the `calc1_val`) and modify the `name`, `source`, and `description` values. Be sure to use a unique **name** for each **ndattribute** in the attributes file.

Note: ndattribute : item specified by an `<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">
        <attribute name="NX_class" source="constant" value="NXdetector" type="string"/>
        <dataset name="data" source="detector" det_default="true">
          <attribute name="NX_class" source="constant" value="SDS" type="string"/>
          <attribute name="signal" source="constant" value="1" type="int"/>
          <attribute name="target" source="constant" value="/entry/instrument/detector/data" type="string"/>
        </dataset>
        <group name="NDAttributes">
          <attribute name="NX_class" source="constant" value="NXcollection" type="string"/>
          <dataset name="ColorMode" source="ndattribute" ndattribute="ColorMode"/>
        </group>          <!-- end group NDAttribute -->
      </group>            <!-- end group detector -->
      <group name="NDAttributes" ndattr_default="true">
        <attribute name="NX_class" source="constant" value="NXcollection" type="string"/>
      </group>            <!-- end group NDAttribute (default) -->
      <group name="performance">
        <dataset name="timestamp" source="ndattribute"/>
      </group>            <!-- end group performance -->
    </group>              <!-- end group instrument -->
    <group name="data">
      <attribute name="NX_class" source="constant" value="NXdata" type="string"/>
      <hardlink name="data" target="/entry/instrument/detector/data"/>
      <!-- The "target" attribute in /entry/instrument/detector/data is used to
           tell Nexus utilities that this is a hardlink -->
    </group>              <!-- end group data -->
  </group>                <!-- end group entry -->
</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"
```

(continues on next page)

(continued from previous page)

```
    type="string"/>
</group>
```

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`:

```
<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:

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

or:

```
<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/

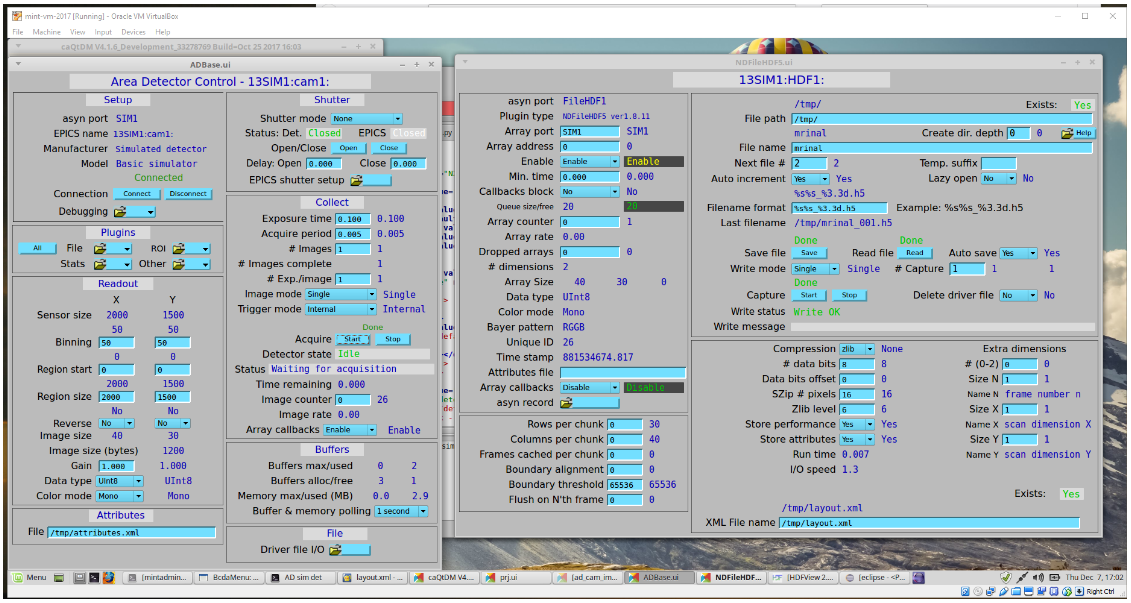


Fig. 14: **ADBase** and **NDFileHDF5** configuration screens

- 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*[7] package to write the HDF5 file.

```
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)

[7] h5py: http://docs.h5py.org

(continued from previous page)

```python
    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)

(continued from previous page)

```
    @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*[8] package for python simplifies the work to create a NeXus file. Rewriting the above code using *nexusformat*:

```
import numpy as np
from nexusformat.nexus import *


def write_nexus_file(fname, image, md={}):
    """
    write the image to a NeXus HDF5 data file

```

(continues on next page)

[8] nexusformat: This Python package is described on the NeXPy web site

(continued from previous page)

```
    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()
    nx['/entry'] = NXentry(NXinstrument(NXdetector()))
    nx['entry/instrument/detector/image'] = NXfield(image, units='counts',
                                                    compression='gzip')
    nx['entry/data'] = NXdata()
    nx['entry/data'].makelink(nx['entry/instrument/detector/image'])
    nx['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[9], nexpy[10], or pymca[11]. Views of the test image shown using **NeXPy** (from the HDF5 file) and **caQtDM** (the image from EPICS) are shown.

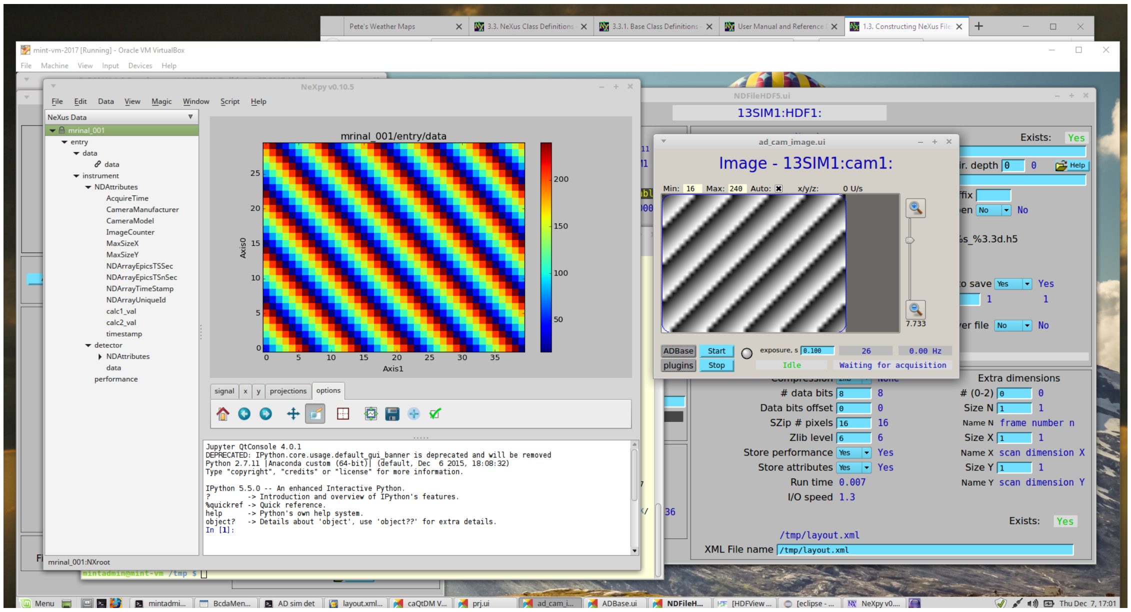


Fig. 15: Views of the image in **NeXPy** (left) and in **caQtDM** (right)

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

file	description
`attributes.xml`	The attributes file
`layout.xml`	The layout file
`mrinal_001.h5`	example NeXus HDF5 file written from EPICS
`write_nexus_file.py`	Python code to get images from EPICS and write a NeXus file
`write_nexus_file2.py`	*write_nexus_file.py* rewritten with *nexusformat* package
`example.h5`	example NeXus HDF5 file written from Python

[9] hdfview: https://support.hdfgroup.org/products/java/hdfview/
[10] nexpy: https://nexpy.github.io/nexpy/
[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.

CHAPTER

THREE

NEXUS: REFERENCE 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 GiHub 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 readbility 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:

  nP: Total number of scan points

  nE: Number of photon energies scanned

  nFrames: Number of frames

  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.
```

(continues on next page)

(continued from previous page)

```
        </doc>
        <group type="NXentry" name="entry">
        <field name="title"  minOccurs="0" maxOccurs="1"/>
...
```

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.

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

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

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.

NXDL element type	minOccurs	maxOccurs
group	[1]	unbounded
field	[1]	unbounded
attribute	[1]	1

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.

Writing references and anchors in the documentation.

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:

```
:ref:`this text will appear <anchor>`
```

[1] For example *XML Copy Editor* (http://xml-copy-editor.sourceforge.net/)

[2] http://en.wikipedia.org/wiki/XML#Well-formedness_and_error-handling

Since these anchors are absolute references, they may be used anywhere in the documentation source (that is, within XML <doc> structures in *.nxdl.xml* files or in `.rst` files).

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*

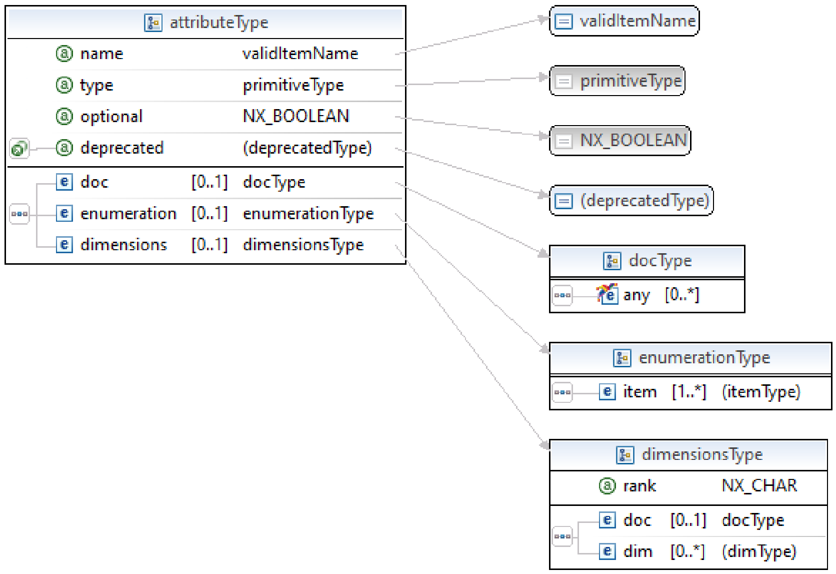


Fig. 1: Graphical representation of the NXDL `attribute` element

dimensions

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*

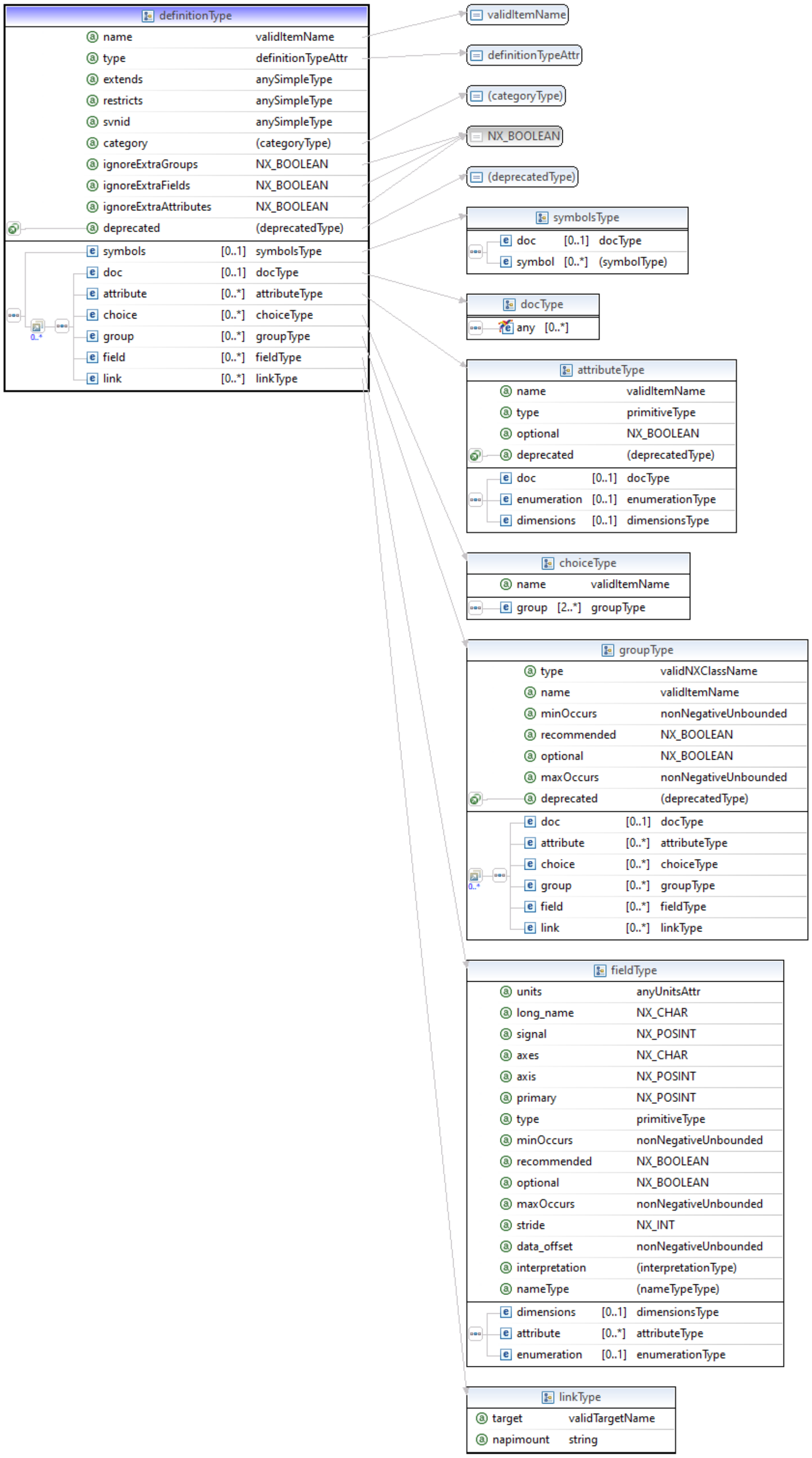


Fig. 2: Graphical representation of the NXDL `definition` element

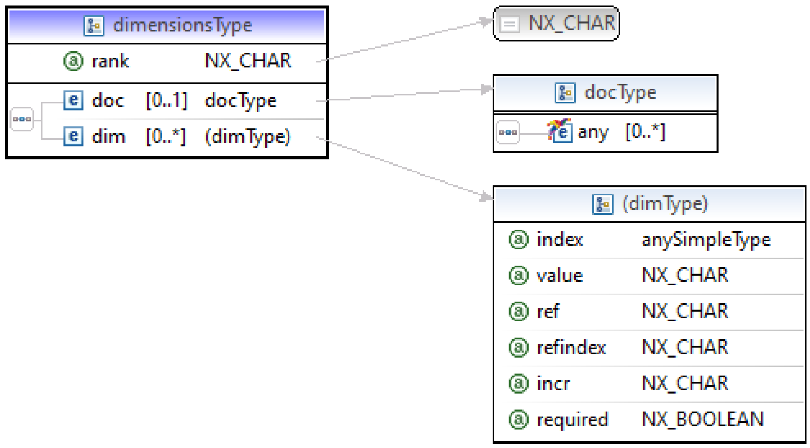


Fig. 3: Graphical representation of the NXDL `dimensions` element

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*

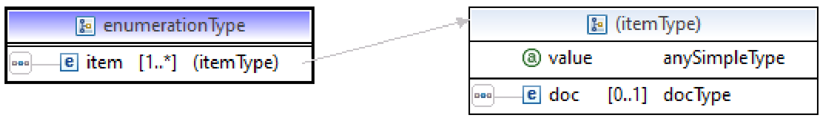


Fig. 5: Graphical representation of the NXDL `enumeration` element

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*

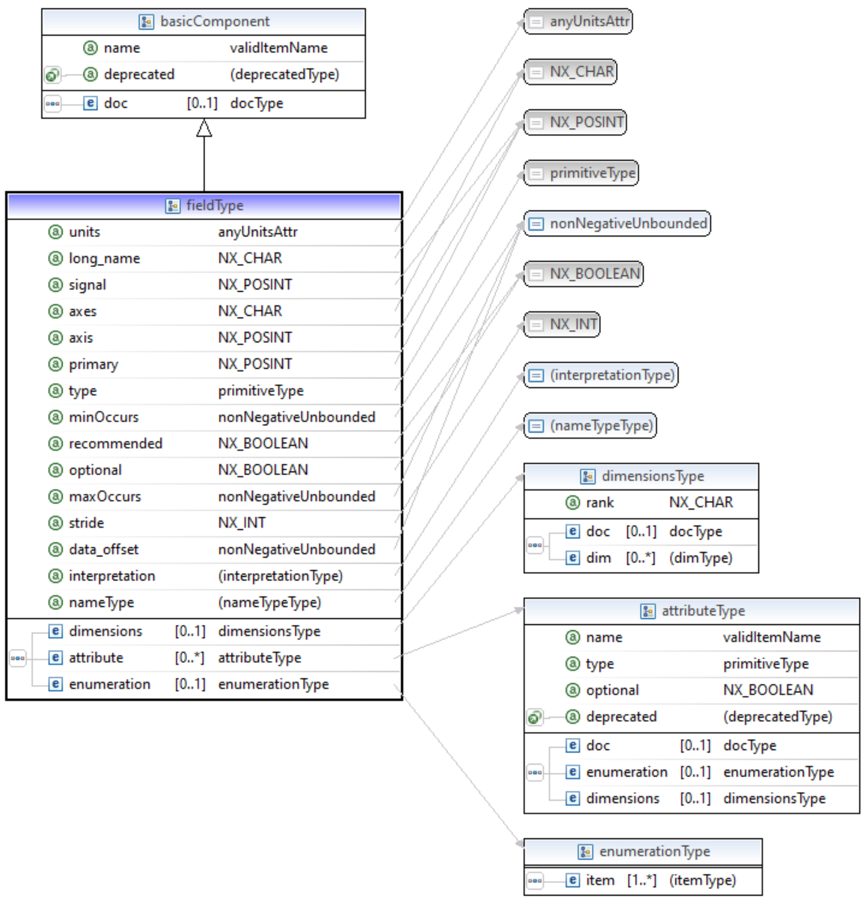


Fig. 6: Graphical representation of the NXDL `field` element

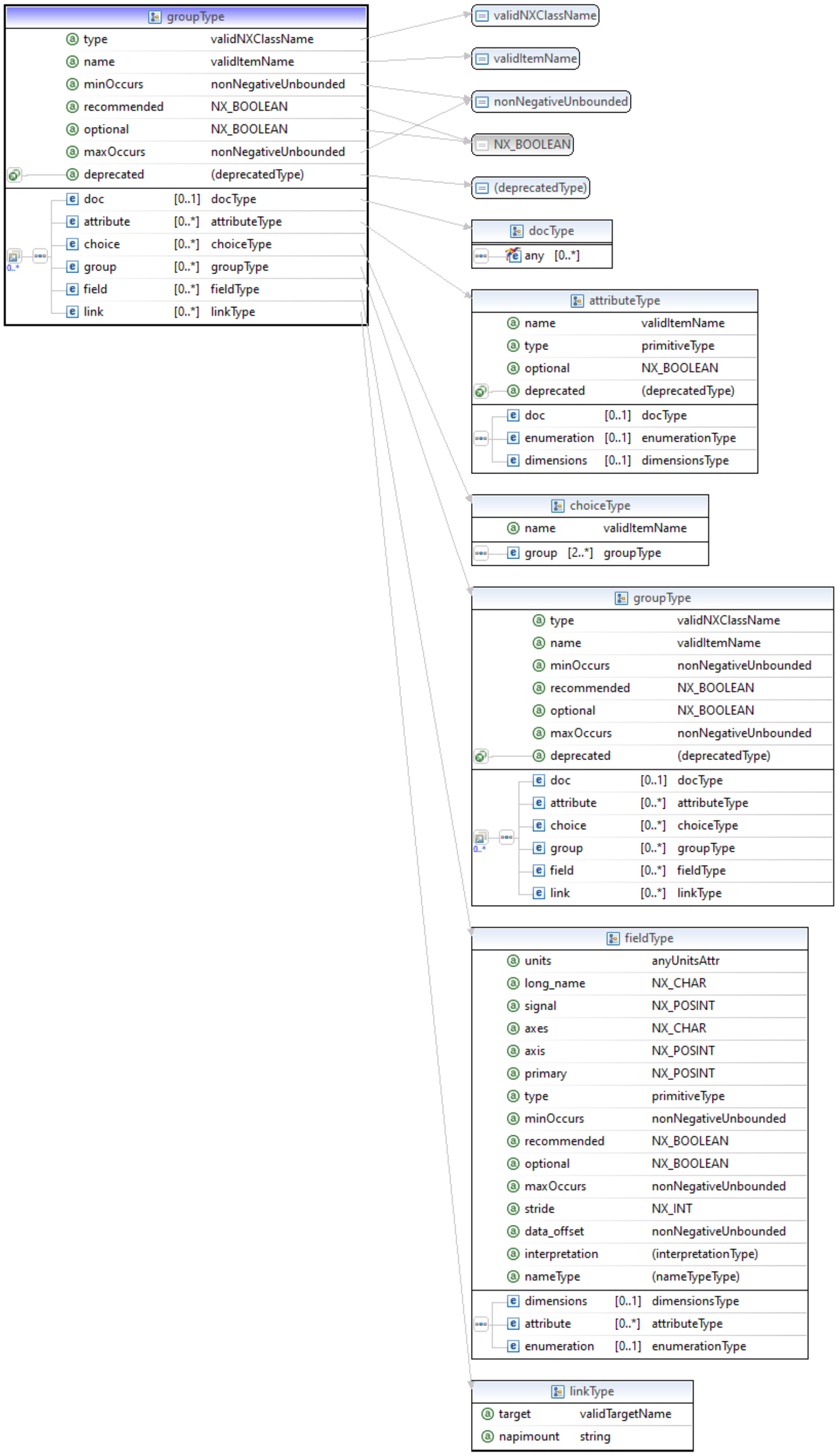


Fig. 7: Graphical representation of the NXDL `group` element

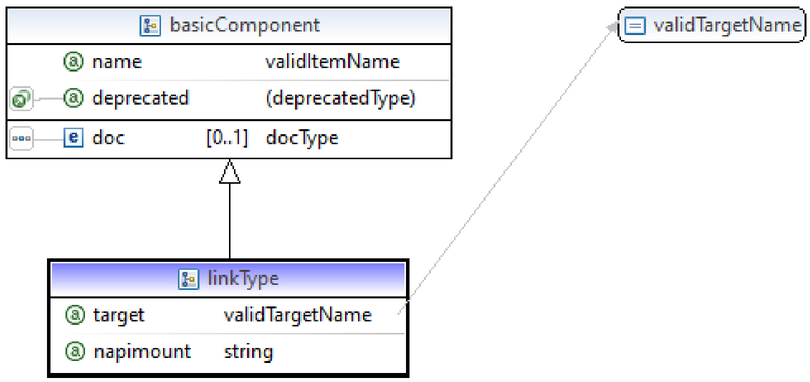


Fig. 8: Graphical representation of the NXDL `link` element

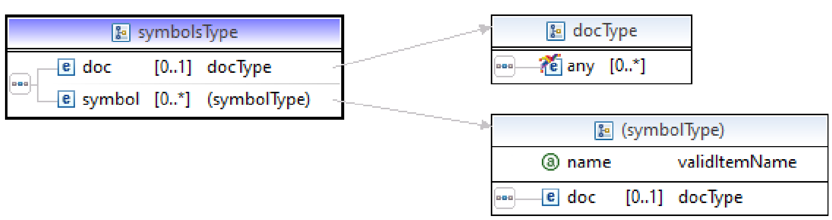


Fig. 9: Graphical representation of the NXDL `symbols` element

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).

@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:

```
Descriptive name of sample
```

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.)

```
source operating mode

        for storage rings
        for storage rings
```

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.

See https://support.hdfgroup.org/HDF5/Tutor/phypereg.html or *4. Dataspace Selection Operations* in https://portal.hdfgroup.org/display/HDF5/Dataspaces

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 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 `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).

See https://support.hdfgroup.org/HDF5/Tutor/phypereg.html or *4. Dataspace Selection Operations* in https://portal.hdfgroup.org/display/HDF5/Dataspaces

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*.

http://manual.nexusformat.org/_static/NeXusIntern.pdf

@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:

```
/: 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:

```
number of reflecting surfaces
number of wavelengths
```

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:

```
(/[a-zA-Z_][\w_]*(:[a-zA-Z_][\w_]*)?)+
```

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 for more details.

The `xs:token` data type
The `xs:string` 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 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_CHAR_OR_NUMBER
Any valid character string or NeXus number representation

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 (`length * angle`) of a radiation source,
example(s): `nm*rad`

NX_ENERGY

units of energy,
example(s): `J | keV`

NX_FLUX

units of flux,
example(s): `1/s/cm^2`

NX_FREQUENCY

units of frequency,
example(s): `Hz`

NX_LENGTH

units of length,
example(s): `m`

NX_MASS

units of mass,
example(s): `g`

NX_MASS_DENSITY

units of mass density,
example(s): `g/cm^3`

NX_MOLECULAR_WEIGHT

units of molecular weight,
example(s): `g/mol`

NX_PERIOD

units of time, period of pulsed source (alias to *NX_TIME*),
example(s): `us`

NX_PER_AREA

units of 1/length^2,
example(s): `1/m^2`

NX_PER_LENGTH

units of 1/length,
example(s): `1/m`

NX_POWER

units of power,
example(s): `W`

NX_PRESSURE

units of pressure,
example(s): `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/nm` | `1/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

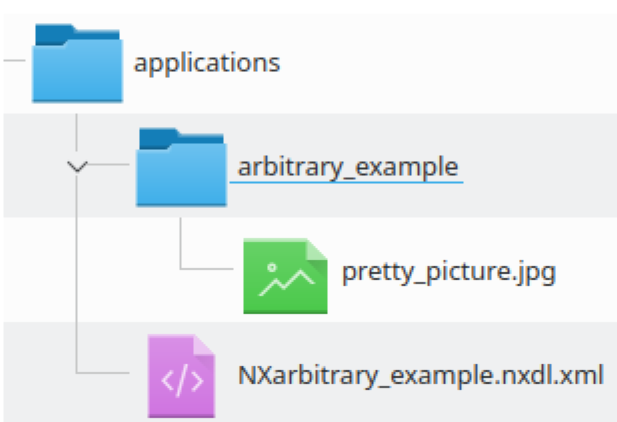
```xml
<definition name="NXarbitrary_example">

<!-- 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`.



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:

```
entry:NXentry
   definition="NXsas"
```

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:

```
entry:NXentry
   raw:NXsubentry
      definition="NXsas"
   reduced:NXsubentry
      definition="NXcanSAS"
   fluo:NXsubentry
      definition="NXfluo"
```

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.

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
The *NXdata* class is designed to encapsulate all the information required for a set of data to be plotted.

NXdetector
A detector, detector bank, or multidetector.

NXdetector_channel
Description and metadata for a single channel from a multi-channel detector.

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.

Note, the group was incorrectly documented as deprecated, but it is not and it is in common use.

Symbols:

No symbol table

Groups cited:
NXgeometry, *NXnote*, *NXoff_geometry*, *NXtransformations*

Structure:

@default: (optional) *NX_CHAR*

Declares which child group contains a path leading to a *NXdata* group.

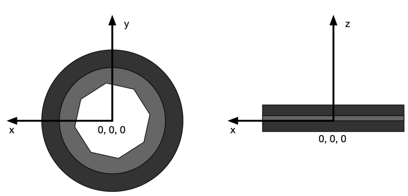
It is recommended (as of NIAC2014) to use this attribute to help define the path to the default dataset to be plotted. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

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 (asymmetic) 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.

OFF_GEOMETRY: (optional) *NXoff_geometry*

Use this group to describe the shape of the aperture

GEOMETRY: (optional) *NXgeometry*

DEPRECATED: Use the field *depends_on* and *NXtransformations* to position the aperture and *NXoff_geometry* to describe its shape

location and shape of aperture

BLADE_GEOMETRY: (optional) *NXgeometry*

DEPRECATED: Use *NXoff_geometry* instead to describe the shape of the aperture

location and shape of each blade

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/NOTE-group*
- */NXaperture/OFF_GEOMETRY-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*

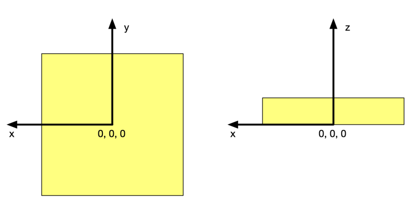
time stamp for this observation

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 (asymmetic) 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. Particulary 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. Finally, There are cases where the beam is considered as a beamline component and this group may be defined as a subgroup directly inside *NXinstrument*, in which case it is recommended that the position of the beam is specified by an *NXtransformations* group, unless the beam is at the origin (which is the sample).

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 presense 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 `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_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_polarization: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [nP, 2]) {units=*NX_ANY*}

Polarization vector on entering beamline component

final_polarization: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [nP, 2]) {units=*NX_ANY*}

Polarization vector on leaving beamline component

incident_polarization_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 NXtransformations 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 rotation.

final_polarization_stokes: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [nP, 4]) {units=*NX_ANY*}

Polarization vector on leaving beamline component using Stokes notation (see incident_polarization_stokes).

final_wavelength_spread: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [m]) {units=*NX_WAVELENGTH*}

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 *The NeXus Coordinate System*). 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@default-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXbeam.nxdl.xml

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.

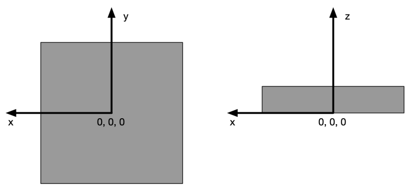
status: (optional) *NX_CHAR*

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/OFF_GEOMETRY-group*
- */NXbending_magnet/source_distance_x-field*
- */NXbending_magnet/source_distance_y-field*

- */NXbending_magnet/spectrum-group*
- */NXbending_magnet/TRANSFORMATIONS-group*
- */NXbending_magnet@default-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXbending_magnet.nxdl.xml

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

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

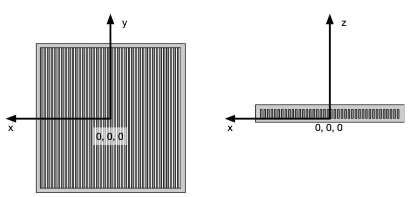
transmitting_material: (optional) *NX_CHAR*

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) *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.

- */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 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 or 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])

Orientation matrix of single crystal sample using Busing-Levy convention: W. R. Busing and H. A. Levy (1967). Acta Cryst. 22, 457-464

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*

- */NXcrystal@default-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXcrystal.nxdl.xml

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:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXcylindrical_geometry.nxdl.xml

NXdata

Status:

base class, extends *NXobject*

Description:

The *NXdata* class is designed to encapsulate all the information required for a set of data to be plotted. NXdata groups contain plottable data (sometimes referred to as *signals* or *dependent variables*) and their associated axis coordinates (sometimes referred to as *axes* or *independent variables*).

The actual names of the *DATA* and *AXISNAME* fields can be chosen *freely*, as indicated by the upper case (this is a common convention in all NeXus classes).

Note: `NXdata` provides data and coordinates to be plotted but does not describe how the data is to be plotted or even the dimensionality of the plot. https://www.nexusformat.org/NIAC2018Minutes.html#nxdata-plottype–attribute

---

Signals:

The *DATA* fields contain the signal values to be plotted. The name of the field to be used as the *default plot signal* is provided by the *signal* attribute. The names of the fields to be used as *secondary plot signals* are provided by the *auxiliary_signals* attribute.

An example with three signals, one of which being the default

```
data:NXdata
  @signal = "data1"
  @auxiliary_signals = ["data2", "data3"]
  data1: float[10,20,30]  --> the default signal
  data2: float[10,20,30]
  data3: float[10,20,30]
```

Axes:

The *AXISNAME* fields contain the axis coordinates associated with the data values. The names of all *AXISNAME* fields are listed in the *axes* attribute.

Rank

AXISNAME fields are typically one-dimensional arrays, which annotate one of the dimensions.

An example of this would be

```
data:NXdata
  @signal = "data"
  @axes = ["x", "y"]  --> the order matters
  data: float[10,20]
  x: float[10]        --> coordinates along the first dimension
  y: float[20]        --> coordinates along the second dimension
```

In this example each data point `data[i,j]` has axis coordinates `[x[i], y[j]]`.

However, the fields can also have a rank greater than 1, in which case the rank of each *AXISNAME* must be equal to the number of data dimensions it spans.

An example of this would be

```
data:NXdata
  @signal = "data"
  @axes = ["x", "y"]  --> the order does NOT matter
  @x_indices = [0, 1]
  @y_indices = [0, 1]
  data: float[10,20]
  x: float[10,20]     --> coordinates along both dimensions
  y: float[10,20]     --> coordinates along both dimensions
```

In this example each data point `data[i,j]` has axis coordinates `[x[i,j], y[i,j]]`.

Dimensions

The data dimensions annotated by an *AXISNAME* field are defined by the *AXISNAME_indices* attribute. When this attribute is missing, the position(s) of the *AXISNAME* string in the *axes* attribute are used.

When all *AXISNAME* fields are one-dimensional, and none of the data dimensions have more than one axis, the *AXISNAME_indices* attributes are often omitted. If one of the data dimensions has no *AXISNAME* field, the string "." can be used in the corresponding index of the axes list.

An example of this would be

```
data:NXdata
  @signal = "data"
  @axes = ["x", ".", "z"]  --> the order matters
  data: float[10,20,30]
  x: float[10]             --> coordinates along the first dimension
  z: float[30]             --> coordinates along the third dimension
```

When using *AXISNAME_indices* this becomes

```
data:NXdata
  @signal = "data"
  @axes = ["x", "z"]       --> the order does NOT matter
  data: float[10,20,30]
  @x_indices = 0
  @z_indices = 2
  x: float[10]             --> coordinates along the first dimension
  z: float[30]             --> coordinates along the third dimension
```

When providing *AXISNAME_indices* attributes it is recommended to do it for all axes.

Non-trivial axes

What follows are two examples where *AXISNAME_indices* attributes cannot be omitted.

The first is an example where data dimensions have alternative axis coordinates. The NXdata group represents a stack of images collected at different energies. The `wavelength` is an alternative axis of `energy` for the last dimension (or vice versa).

```
data:NXdata
  @signal = "data"
  @axes = ["x", "y", "energy", "wavelength"]  --> the order does NOT matter
  @x_indices = 0
  @y_indices = 1
  @energy_indices = 2
  @wavelength_indices = 2
  data: float[10,20,30]
  x: float[10]              --> coordinates along the first dimension
  y: float[20]              --> coordinates along the second dimension
  energy: float[30]         --> coordinates along the third dimension
  wavelength: float[30]     --> coordinates along the third dimension
```

The second is an example with coordinates that span more than one dimension. The NXdata group represents data from 2D mesh scans performed at multiple energies. Each data point `data[i,j,k]` has axis coordinates `[x[i,j,k], y[i,j,k], energy[k]]`.

```
data:NXdata
  @signal = "data"
  @axes = ["x", "y", "energy"]  --> the order does NOT matter
  @x_indices = [0, 1, 2]
  @y_indices = [0, 1, 2]
  @energy_indices = 2
  data: float[10,20,30]
  x: float[10,20,30]            --> coordinates along all dimensions
  y: float[10,20,30]            --> coordinates along all dimensions
  energy: float[30]             --> coordinates along the third dimension
```

Uncertainties:

Standard deviations on data values as well as coordinates can be provided by *FIELDNAME_errors* fields where `FIELDNAME` is the name of a *DATA* field or an *AXISNAME* field.

An example of uncertainties on the signal, auxiliary signals and axis coordinates

```
data:NXdata
  @signal = "data1"
  @auxiliary_signals = ["data2", "data3"]
  @axes = ["x", "z"]
  @x_indices = 0
  @z_indices = 2
  data1: float[10,20,30]
  data2: float[10,20,30]
  data3: float[10,20,30]
  x: float[10]
  z: float[30]
  data1_errors: float[10,20,30]
  data2_errors: float[10,20,30]
  data3_errors: float[10,20,30]
```

(continues on next page)

(continued from previous page)

```
x_errors: float[10]
z_errors: float[30]
```

Symbols:

These symbols will be used below to coordinate fields with the same shape.

dataRank: rank of the `DATA` field(s)

nx: length of the `x` field

ny: length of the `y` field

nz: length of the `z` field

Groups cited:
none

Structure:

@signal: (optional) *NX_CHAR*

The value is the *name* of the signal that contains the default plottable data. This field or link *must* exist and be a direct child of this NXdata 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.

@auxiliary_signals: (optional) *NX_CHAR*

Array of strings holding the *names* of additional signals to be plotted with the *default signal*. These fields or links *must* exist and be direct children of this NXdata group.

Each auxiliary signal needs to be of the same shape as the default signal.

@default_slice: (optional) *NX_CHAR_OR_NUMBER*

Which slice of data to show in a plot by default. This is useful especially for datasets with more than 2 dimensions.

Should be an array of length equal to the number of dimensions in the data, with the following possible values:

- ".": All the data in this dimension should be included
- Integer: Only this slice should be used.
- String: Only this slice should be used. Use if `AXISNAME` is a string array.

Example:

```
data:NXdata
      @signal = "data"
      @axes = ["image_id", "channel", ".", "."]
      @image_id_indices = 0
      @channel_indices = 1
      @default_slice = [".", "difference", ".", "."]
      image_id = [1, ..., nP]
      channel = ["threshold_1", "threshold_2", "difference"]
      data = uint[nP, nC, i, j]
```

Here, a data array with four dimensions, including the number of images (nP) and number of channels (nC), specifies more dimensions than can be visualized with a 2D image viewer for a given image. Therefore the default_slice attribute specifies that the "difference" channel should be shown by default.

Alternate version using an integer would look like this (note 2 is a string):

```
data:NXdata
      @signal = "data"
      @axes = ["image_id", "channel", ".", "."]
      @image_id_indices = 0
      @channel_indices = 1
      @default_slice = [".", "2", ".", "."]
      image_id = [1, ..., nP]
      channel = ["threshold_1", "threshold_2", "difference"]
      data = uint[nP, nC, i, j]
```

@AXISNAME_indices: (optional) *NX_INT*

The `AXISNAME_indices` attribute is a single integer or an array of integers that defines which *data* dimension(s) are spanned by the corresponding axis. The first dimension index is `0` (zero).

When the `AXISNAME_indices` attribute is missing for an *AXISNAME* field, its value becomes the index (or indices) of the *AXISNAME* name in the *axes* attribute.

Note: When `AXISNAME_indices` contains multiple integers, it must be saved as an actual array of integers and not a comma separated string.

@axes: (optional) *NX_CHAR*

The `axes` attribute is a list of strings which are the names of the *AXISNAME* fields that contain the values of the coordinates along the *data* dimensions.

Note: When `axes` contains multiple strings, it must be saved as an actual array of strings and not a single comma separated string.

AXISNAME: (optional) *NX_CHAR_OR_NUMBER*

Coordinate values along one or more *data* dimensions. The rank must be equal to the number of dimensions it spans.

As the upper case `AXISNAME` indicates, the names of the `AXISNAME` fields can be chosen *freely*. The *axes* attribute can be used to find all datasets in the `NXdata` that contain coordinate values.

Most AXISNAME fields will be sequences of numbers but if an axis is better represented using names, such as channel names, an array of NX_CHAR can be provided. **@long_name**: (optional) *NX_CHAR*

Axis label

@units: (optional) *NX_CHAR*

Unit in which the coordinate values are expressed. See the section *NeXus Data Units* for more information.

@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* attribute is now preferred.

DATA: (optional) *NX_NUMBER* (Rank: dataRank)

Data values to be used as the NeXus *plottable data*. As the upper case `DATA` indicates, the names of the `DATA` fields can be chosen *freely*. The *signal attribute* and *auxiliary_signals attribute* can be used to find all datasets in the `NXdata` that contain data values.

The maximum rank is `32` for compatibility with backend file formats. **@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 coordinates (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

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.

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.

This is a special case of a *AXISNAME field* kept for backward compatiblity.

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.

This is a special case of a *AXISNAME field* kept for backward compatiblity.

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.

This is a special case of a *AXISNAME field* kept for backward compatiblity.

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/AXISNAME@units-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@default_slice-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_channel*, *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.

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*

y-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 allow to distinguish different types of exposure to the same detector "data" field. Some techniques require frequent (re-)calibration inbetween measuremnts 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 NXoff_geometry 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 NXoff_geometry to describe its shape instead

Position and orientation of detector

CHANNELNAME_channel: (optional) *NXdetector_channel*

Group containing the description and metadata for a single channel from a multi-channel detector.

Given an *NXdata* group linked as part of an NXdetector group that has an axis with named channels (see the example in *NXdata*), the NXdetector will have a series of NXdetector_channel groups, one for each channel, named CHANNELNAME_channel.

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/CHANNELNAME_channel-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/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_channel

Status:

base class, extends *NXobject*

Description:

Description and metadata for a single channel from a multi-channel detector.

Given an *NXdata* group linked as part of an NXdetector group that has an axis with named channels (see the example in *NXdata*), the NXdetector will have a series of NXdetector_channel groups, one for each channel, named CHANNELNAME_channel.

Example, given these axes in the NXdata group:

```
@axes = ["image_id", "channel", ".", "."]
```

And this list of channels in the NXdata group:

```
channel = ["threshold_1", "threshold_2", "difference"]
```

The NXdetector group would have three NXdetector_channel groups:

```
detector:NXdetector
  ...
  threshold_1_channel:NXdetector_channel
    threshold_energy = float
    flatfield = float[i, j]
    pixel_mask = uint[i, j]
    flatfield_applied = bool
    pixel_mask_applied = bool
  threshold_2_channel:NXdetector_channel
    threshold_energy = float
    flatfield = float[i, j]
    pixel_mask = uint[i, j]
    flatfield_applied = bool
    pixel_mask_applied = bool
  difference_channel:NXdetector_channel
    threshold_energy = float[2]
```

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 (2D), but three or higher dimensional arrays can be produced by a detector at each trigger.

dataRank: Rank of the `data` field associated with this detector

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

Groups cited:
none

Structure:

threshold_energy: (optional) *NX_FLOAT* {units=*NX_ENERGY*}

Energy at which a photon will be recorded

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]])

Response of each pixel given a constant input

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: dataRank, Dimensions: [., i, j, [k]])

Custom pixel mask for this channel. May include nP as the first dimension for masks that vary for each scan point.

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.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXdetector_channel/flatfield-field*
- */NXdetector_channel/flatfield_applied-field*
- */NXdetector_channel/flatfield_errors-field*
- */NXdetector_channel/pixel_mask-field*
- */NXdetector_channel/pixel_mask_applied-field*
- */NXdetector_channel/saturation_value-field*
- */NXdetector_channel/threshold_energy-field*
- */NXdetector_channel/underload_value-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXdetector_channel.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:

```
DTL    DTR
DLL    DLR
```

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 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*}

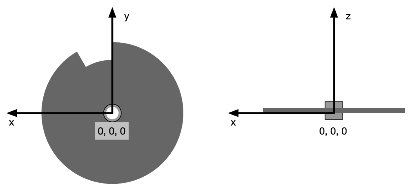
Low and high values of wavelength range transmitted

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.



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*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/base_classes/NXdisk_chopper.nxdl.xml

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.

entry_identifier: (optional) *NX_CHAR*

unique identifier for the measurement, defined by the facility.

entry_identifier_uuid: (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

end_time: (optional) *NX_DATE_TIME*

Ending 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 definition 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 extensive processing.

Certain application definitions override this decision and require an *NXdata* group in the *NXentry* 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*

[1] NIAC2016: https://www.nexusformat.org/NIAC2016.html, https://github.com/nexusformat/NIAC/issues/16

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

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/NOTE-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 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 alphabetic 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])

Orientation matrix of single crystal filter using Busing-Levy convention: W. R. Busing and H. A. Levy (1967). Acta Cryst. 22, 457-464

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.

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*

- */NXgrating/figure_data-group*
- */NXgrating/interior_atmosphere-field*
- */NXgrating/layer_thickness-field*
- */NXgrating/OFF_GEOMETRY-group*
- */NXgrating/period-field*
- */NXgrating/shape-group*
- */NXgrating/substrate_density-field*
- */NXgrating/substrate_material-field*
- */NXgrating/substrate_roughness-field*
- */NXgrating/substrate_thickness-field*
- */NXgrating/TRANSFORMATIONS-group*
- */NXgrating@default-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXgrating.nxdl.xml

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) *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*

A description of this particular instance of `NXguide`.

incident_angle: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

TODO: documentation needed

bend_angle_x: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

TODO: documentation needed

bend_angle_y: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

TODO: documentation needed

interior_atmosphere: (optional) *NX_CHAR*

Any of these values: `vacuum` | `helium` | `argon`

external_material: (optional) *NX_CHAR*

external material outside substrate

m_value: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [nsurf])

The `m` value for a supermirror, which defines the supermirror regime in multiples of the critical angle of Nickel.

substrate_material: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [nsurf])

TODO: documentation needed

substrate_thickness: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [nsurf]) {units=*NX_LENGTH*}

TODO: documentation needed

coating_material: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [nsurf])

TODO: documentation needed

substrate_roughness: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [nsurf]) {units=*NX_LENGTH*}

TODO: documentation needed

coating_roughness: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [nsurf]) {units=*NX_LENGTH*}

TODO: documentation needed

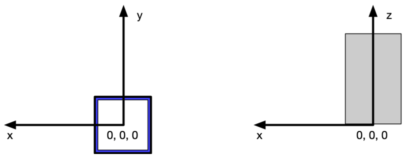
number_sections: (optional) *NX_INT* {units=*NX_UNITLESS*}

number of substrate sections (also called `nsurf` as an index in the `NXguide` specification)

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 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) *NXgeometry*

DEPRECATED: Use the field *depends_on* and *NXtransformations* to position the guid and NXoff_geometry to describe its shape instead

TODO: Explain what this NXgeometry group means. What is intended here?

reflectivity: (optional) *NXdata*

Reflectivity as function of reflecting surface and wavelength

@signal: (optional) *NX_CHAR* *<=*

Obligatory value: `data`

@axes: (optional) *NX_CHAR* *<=*

Obligatory value: `surface wavelength`

@surface_indices: (optional) *NX_CHAR*

Obligatory value: `0`

@wavelength_indices: (optional) *NX_CHAR*

Obligatory value: `1`

data: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [nsurf, nwl]) *<=*

reflectivity of each surface as a function of wavelength

surface: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [nsurf]) {units=*NX_ANY*}

List of surfaces. Probably best to use index numbers but the specification is very loose.

wavelength: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [nwl]) {units=*NX_WAVELENGTH*}

wavelengths at which reflectivity was measured

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.

- */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*

- */NXinsertion_device@default-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXinsertion_device.nxdl.xml

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*, *NXpositioner*, *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 accomodate non standard clocks.

This method of storing logged data helps to distinguish instances in which a variable contains signal or axis coordinate values of plottable 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".

@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) *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:

- `single`: mirror with a single material as a reflecting surface
- `multi`: mirror with stacked, multiple layers as a reflecting surface

description: (optional) *NX_CHAR*

description of this mirror

incident_angle: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

bend_angle_x: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

bend_angle_y: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

interior_atmosphere: (optional) *NX_CHAR*

Any of these values: `vacuum` | `helium` | `argon`

external_material: (optional) *NX_CHAR*

external material outside substrate

m_value: (optional) *NX_FLOAT* {units=*NX_UNITLESS*}

The m value for a supermirror, which defines the supermirror regime in multiples of the critical angle of Nickel.

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*}

even_layer_material: (optional) *NX_CHAR*

even_layer_density: (optional) *NX_FLOAT* {units=*NX_MASS_DENSITY*}

odd_layer_material: (optional) *NX_CHAR*

odd_layer_density: (optional) *NX_FLOAT* {units=*NX_MASS_DENSITY*}

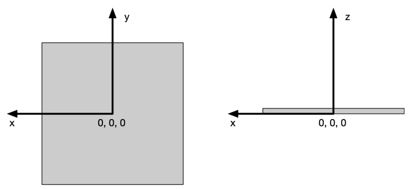
layer_thickness: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

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.

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.



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/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 *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*}

Effective distance as seen by measuring radiation. Note, it is recommended to use NXtransformations instead.

type: (optional) *NX_CHAR*

Any of these values:

- H2O
- D2O
- Liquid H2
- Liquid CH4
- Liquid D2
- Solid D2
- C
- Solid CH4
- Solid H2

poison_depth: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

coupled: (optional) *NX_BOOLEAN*

whether the moderator is coupled

coupling_material: (optional) *NX_CHAR*

The material used for coupling. Usually Cd.

poison_material: (optional) *NX_CHAR*

Any of these values: Gd | Cd

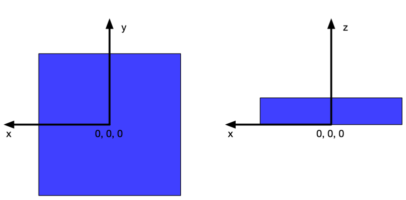
temperature: (optional) *NX_FLOAT* {units=*NX_TEMPERATURE*}

average/nominal moderator 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.

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 axis coordinates, 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 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

end_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)

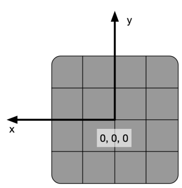
count_time: (optional) *NX_FLOAT* {units=*NX_TIME*}

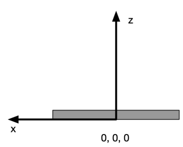
Elapsed actual counting time, can be an array of size `np` when scanning. This is not the difference of the calendar time but the time the instrument was really counting, without pauses or times lost due beam unavailability

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*

geometry: (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:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXmonochromator.nxdl.xml

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.

value: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [numobj, 6]) {units=*NX_UNITLESS*}

The orientation information is stored as direction cosines. The direction cosines will be between the local coordinate directions and the reference directions (to origin or relative NXgeometry). Calling the local unit vectors (x',y',z') and the reference unit vectors (x,y,z) the six numbers will be [x' dot x, x' dot y, x' dot z, y' dot x, y' dot y, y' dot z] where "dot" is the scalar dot product (cosine of the angle between the unit vectors). The unit vectors in both the local and reference coordinates are right-handed and orthonormal.

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 beused for the corresponding fields.

A PDB entry is unambigous 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_cbfsf" 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 attribues is:

```
(datablock1):NXpdb
   @NXpdb_class:CBF_cbfdb
   (category1):NXpdb
     @NXpdb_class:CBF_cbfcat
      (column_name1):[...]
      (column_name2):[...]
      (column_name3):[...]
      ...
   (category2):NXpdb
```

(continues on next page)

(continued from previous page)

```
      @NXpdb_class:CBF_cbfcat
      (column_name4):[...]
      (column_name5):[...]
      (column_name6):[...]
      ...
   (saveframe1):NXpdb
      @NXpdb_class:CBF_cbfsf
      (category3):NXpdb
        @NXpdb_class:CBF_cbfcat
        (column_name7):[...]
        (column_name8):[...]
        (column_name9):[...]
         ...
      ...
   ...
```

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
```

(continues on next page)

(continued from previous page)

```
_entity_poly.nstd_linkage
_entity_poly.nstd_monomer
_entity_poly.pdbx_seq_one_letter_code
_entity_poly.pdbx_seq_one_letter_code_can
_entity_poly.type
1 no no GIVEQCCTSICSLYQLENYCN GIVEQCCTSICSLYQLENYCN polypeptide(L)
2 no no FVNQHLCGSHLVEALYLVCGERGFFYTPKA FVNQHLCGSHLVEALYLVCGERGFFYTPKA
polypeptide(L)
```

which converts to:

```
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*

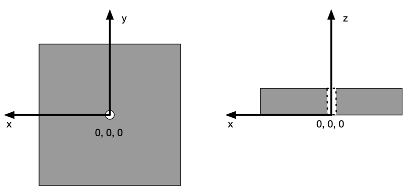
Declares which child group contains a path leading to a *NXdata* group.

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) *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.

- */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 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 …

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:

bit	name
0	predicted
1	observed
2	indexed
3	used_in_refinement
4	strong
5	reference_spot
6	dont_integrate
7	integrated_sum
8	integrated_prf
9	integrated
10	overloaded
11	overlapped
12	overlapped_fg
13	in_powder_ring
14	foreground_includes_bad_pixels
15	background_includes_bad_pixels
16	includes_bad_pixels
17	bad_shoebox
18	bad_spot
19	used_in_modelling
20	centroid_outlier
21	failed_during_background_modelling
22	failed_during_summation
23	failed_during_profile_fitting
24	bad_reference

@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 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 recorded reflection. Should be an integer array of length 6, where the 6 values are pixel positions or frame numbers, as follows:

index	meaning
0	The lower pixel x position
1	The upper pixel x position
2	The lower pixel y position
3	The upper pixel y position
4	The lower frame number
5	The upper frame number

@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*

entries

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*}

This will follow the Busing-Levy convention: W. R. Busing and H. A. Levy (1967). Acta Cryst. 22, 457-464

orientation_matrix: (optional) *NX_FLOAT* (Rank: 3, Dimensions: [n_comp, 3, 3])

Orientation matrix of single crystal sample using Busing-Levy convention: W. R. Busing and H. A. Levy (1967). Acta Cryst. 22, 457-464

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 perferred 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/scattering_length_density-field*
- */NXsample/short_title-field*
- */NXsample/situation-field*
- */NXsample/space_group-field*
- */NXsample/stress_field-field*
- */NXsample/stress_field@direction-attribute*
- */NXsample/temperature-field*
- */NXsample/temperature_env-group*
- */NXsample/temperature_log-group*
- */NXsample/thickness-field*
- */NXsample/TRANSFORMATIONS-group*
- */NXsample/transmission-group*
- */NXsample/type-field*
- */NXsample/ub_matrix-field*
- */NXsample/unit_cell-field*
- */NXsample/unit_cell_abc-field*
- */NXsample/unit_cell_alphabetagamma-field*
- */NXsample/unit_cell_class-field*
- */NXsample/unit_cell_volume-field*
- */NXsample/volume_fraction-field*
- */NXsample/x_translation-field*
- */NXsample@default-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXsample.nxdl.xml

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 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*}

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*
- */NXsample_component/point_group-field*
- */NXsample_component/relative_molecular_mass-field*
- */NXsample_component/sample_orientation-field*
- */NXsample_component/scattering_length_density-field*
- */NXsample_component/space_group-field*
- */NXsample_component/transmission-group*
- */NXsample_component/unit_cell_abc-field*
- */NXsample_component/unit_cell_alphabetagamma-field*
- */NXsample_component/unit_cell_class-field*
- */NXsample_component/unit_cell_volume-field*
- */NXsample_component/volume_fraction-field*
- */NXsample_component@default-attribute*

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 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`
- `nxpolynomial`

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
- 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 nxpolynomial, 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*

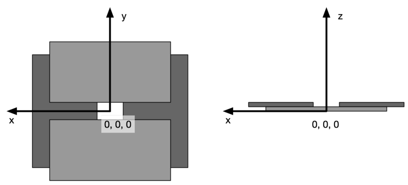
Declares which child group contains a path leading to a *NXdata* group.

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.



x_gap: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Size of the gap opening in the first dimension of the local coordinate system.

y_gap: (optional) *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*

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*}

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) *NX_FLOAT* {units=*NX_POWER*}

Source power

emittance_x: (optional) *NX_FLOAT* {units=*NX_EMITTANCE*}

Source emittance (nm-rad) in X (horizontal) direction.

emittance_y: (optional) *NX_FLOAT* {units=*NX_EMITTANCE*}

Source emittance (nm-rad) in Y (horizontal) direction.

sigma_x: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

particle beam size in x

sigma_y: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

particle beam size in y

flux: (optional) *NX_FLOAT* {units=*NX_FLUX*}

Source intensity/area (example: s-1 cm-2)

energy: (optional) *NX_FLOAT* {units=*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) *NX_FLOAT* {units=*NX_CURRENT*}

Accelerator, X-ray tube, or storage ring current

voltage: (optional) *NX_FLOAT* {units=*NX_VOLTAGE*}

Accelerator voltage

frequency: (optional) *NX_FLOAT* {units=*NX_FREQUENCY*}

Frequency of pulsed source

period: (optional) *NX_FLOAT* {units=*NX_PERIOD*}

Period of pulsed source

target_material: (optional) *NX_CHAR*

Pulsed source target material

Any of these values:

- `Ta`
- `W`
- `depleted_U`
- `enriched_U`
- `Hg`
- `Pb`
- `C`

number_of_bunches: (optional) *NX_INT*

For storage rings, the number of bunches in use.

bunch_length: (optional) *NX_FLOAT* {units=*NX_TIME*}

For storage rings, temporal length of the bunch

bunch_distance: (optional) *NX_FLOAT* {units=*NX_TIME*}

For storage rings, time between bunches

pulse_width: (optional) *NX_FLOAT* {units=*NX_TIME*}

temporal width of source pulse

mode: (optional) *NX_CHAR*

source operating mode

Any of these values:

- `Single Bunch`: for storage rings
- `Multi Bunch`: for storage rings

top_up: (optional) *NX_BOOLEAN*

Is the synchrotron operating in top_up mode?

last_fill: (optional) *NX_NUMBER* {units=*NX_CURRENT*}

For storage rings, the current at the end of the most recent injection.

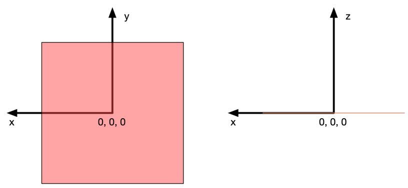
@time: (optional) *NX_DATE_TIME*

date and time of the most recent injection.

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 source plane is its center in the x and y axis. The source is considered infinitely thin in the z axis.



notes: (optional) *NXnote*

any source/facility related messages/events that occurred during the experiment

bunch_pattern: (optional) *NXdata*

For storage rings, description of the bunch pattern. This is useful to describe irregular bunch patterns. **title**: (optional) *NX_CHAR* <=

name of the bunch pattern

pulse_shape: (optional) *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*
- */NXsource/period-field*
- */NXsource/power-field*
- */NXsource/probe-field*
- */NXsource/pulse_shape-group*
- */NXsource/pulse_width-field*
- */NXsource/sigma_x-field*
- */NXsource/sigma_y-field*
- */NXsource/target_material-field*
- */NXsource/top_up-field*
- */NXsource/TRANSFORMATIONS-group*
- */NXsource/type-field*
- */NXsource/voltage-field*
- */NXsource@default-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXsource.nxdl.xml

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 *NXdata* group contains the data to be shown by default. It is used to resolve ambiguity when one *NXdata* 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.

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.

entry_identifier: (optional) *NX_CHAR*

unique identifier for the measurement, defined by the facility.

definition: (optional) *NX_CHAR*

Official NeXus NXDL schema to which this subentry conforms

@version: (optional) *NX_CHAR*

NXDL version number

@URL: (optional) *NX_CHAR*

URL of NXDL file

definition_local: (optional) *NX_CHAR*

Local NXDL schema extended from the subentry specified in the `definition` field. This contains any locally-defined, additional fields in the subentry. **@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

end_time: (optional) *NX_DATE_TIME*

Ending 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 definition 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.

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. **@mime_type**: (optional) *NX_CHAR*

The value should be an `image/*`

Obligatory value: `image/*`

USER: (optional) *NXuser*

SAMPLE: (optional) *NXsample*

INSTRUMENT: (optional) *NXinstrument*

COLLECTION: (optional) *NXcollection*

MONITOR: (optional) *NXmonitor*

DATA: (optional) *NXdata*

PARAMETERS: (optional) *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*
- */NXsubentry/thumbnail@mime_type-attribute*
- */NXsubentry/title-field*
- */NXsubentry/USER-group*
- */NXsubentry@default-attribute*
- */NXsubentry@IDF_Version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXsubentry.nxdl.xml

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 dscription 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 atttributes which are defined through the name are spcified. 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. In this case the value of the `AXISNAME` field is not used and can be set to the number `NaN`.

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.

geometry: (optional) *NXgeometry*

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

fax_number: (optional) *NX_CHAR*

Fax number of user

email: (optional) *NX_CHAR*

Email of user

facility_user_id: (optional) *NX_CHAR*

facility based unique identifier for this person e.g. their identification code on the facility address/contact database

ORCID: (optional) *NX_CHAR*

an author code, Open Researcher and Contributor ID, defined by https://orcid.org and expressed as a URI

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*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXuser.nxdl.xml

NXvelocity_selector

Status:

base class, extends *NXobject*

Description:

A neutron velocity selector

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 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/).

NXarpes
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 a 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 a 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 burocracy 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*

- */NXarpes/ENTRY/INSTRUMENT/analyser/energies-field*
- */NXarpes/ENTRY/INSTRUMENT/analyser/entrance_slit_setting-field*
- */NXarpes/ENTRY/INSTRUMENT/analyser/entrance_slit_shape-field*
- */NXarpes/ENTRY/INSTRUMENT/analyser/entrance_slit_size-field*
- */NXarpes/ENTRY/INSTRUMENT/analyser/lens_mode-field*
- */NXarpes/ENTRY/INSTRUMENT/analyser/pass_energy-field*
- */NXarpes/ENTRY/INSTRUMENT/analyser/region_origin-field*
- */NXarpes/ENTRY/INSTRUMENT/analyser/region_size-field*
- */NXarpes/ENTRY/INSTRUMENT/analyser/sensor_size-field*
- */NXarpes/ENTRY/INSTRUMENT/analyser/time_per_channel-field*
- */NXarpes/ENTRY/INSTRUMENT/monochromator-group*
- */NXarpes/ENTRY/INSTRUMENT/monochromator/energy-field*
- */NXarpes/ENTRY/INSTRUMENT/SOURCE-group*
- */NXarpes/ENTRY/INSTRUMENT/SOURCE/name-field*
- */NXarpes/ENTRY/INSTRUMENT/SOURCE/probe-field*
- */NXarpes/ENTRY/INSTRUMENT/SOURCE/type-field*
- */NXarpes/ENTRY/SAMPLE-group*
- */NXarpes/ENTRY/SAMPLE/name-field*
- */NXarpes/ENTRY/SAMPLE/temperature-field*
- */NXarpes/ENTRY/start_time-field*
- */NXarpes/ENTRY/title-field*
- */NXarpes/ENTRY@entry-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXarpes.nxdl.xml

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/
- http://cansas-org.github.io/canSAS2012/
- 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:

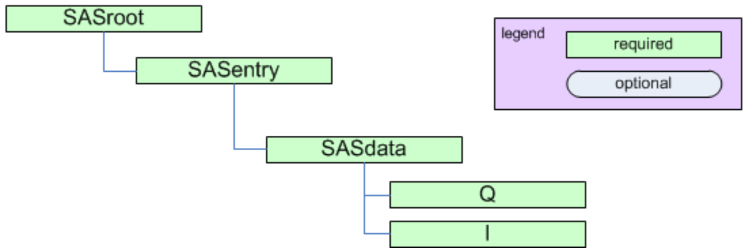


Fig. 10: The minimum requirements for *reduced* small-angle scattering data. (`full image`) See *below* for the minimum required information for a NeXus data file written to the NXcanSAS specification.

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 $I(\vec{Q})$ or $I(|\vec{Q}|)$
- 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.

group (*)	NX_class	canSAS_class
sasentry	NXentry	SASentry
sasdata	NXdata	SASdata
sasdetector	NXdetector	SASdetector
sasinstrument	NXinstrument	SASinstrument
sasnote	NXnote	SASnote
sasprocess	NXprocess	SASprocess
sasprocessnote	NXcollection	SASprocessnote
sastransmission	NXdata	SAStransmission_spectrum
sassample	NXsample	SASsample
sassource	NXsource	SASsource

(*) 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.

```
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[1] 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:

```
@version="1.1"
```

[1] The UDunits specification also includes instructions for derived units.

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 en 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, $|\vec{Q}|$.

$$|\vec{Q}| = (4\pi/\lambda)sin(\theta)$$

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*

[2] ISO-8601 standard time representation.
NeXus dates and times are reported in ISO-8601 (e.g., `yyyy-mm-ddThh: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.

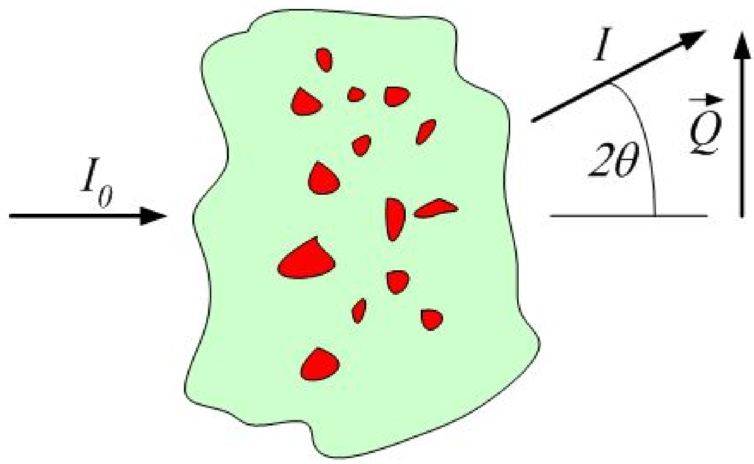


Fig. 11: The $\vec{Q}$ geometry. (`full image`)

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/angstrom`

@uncertainties: (optional) *NX_CHAR*

(optional: for numerical arrays)

Names the dataset (in this SASdata 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 *SASdata* 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 (*dQw*, *dQl*), use a string array, such as:

```
@resolutions="dQw", "dQl"
```

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 ~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 SASdata 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 *SASdata* 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
- `m2/g`
- `cm2/g`
- `arbitrary`

Qdev: (optional) *NX_NUMBER* {units=*NX_PER_LENGTH*}

Estimated Q **resolution** (usually standard deviation). 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="Qdev"
```

or:

```
Q/@resolutions="dQw", "dQl"
```

@units: (required) *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`

dQw: (optional) *NX_NUMBER* {units=*NX_PER_LENGTH*}

Q **resolution** along the axis of scanning (the high-resolution *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 *resolutions* attribute of Q, as in:

```
Q/@resolutions="dQw", "dQl"
```

@units: (required) *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`

dQl: (optional) *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) *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`

Qmean: (optional) *NX_NUMBER* {units=*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 `Qmean` will have the same units. **@units**: (required) *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`

ShadowFactor: (optional) *NX_CHAR* {units=*NX_DIMENSIONLESS*}

A numerical factor applied to pixels affected by the beam stop penumbra. Used in data files from NIST/NCNR instruments.

See: J.G. Barker and J.S. Pedersen (1995) *J. Appl. Cryst.* **28**, 105-114.

INSTRUMENT: (optional) *NXinstrument <=*

Description of the small-angle scattering instrument.

Consider, carefully, the relevance to the SAS data analysis process when adding subgroups in this **NXinstrument** group. Additional information can be added but will likely be ignored by standardized data anlysis processes.

The NeXus *NXbeam* base class may be added as a subgroup of this **NXinstrument** group *or* as a subgroup of the **NXsample** group to describe properties of the beam at any point downstream from the source. **@canSAS_class**: (required) *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) *NX_NUMBER* {units=*NX_PER_LENGTH*}

Slit length of the instrument for this detector, expressed in the same units as Q.

x_position: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Location of the detector in x

y_position: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Location of the detector in y

roll: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Rotation of the detector about the z axis (roll)

pitch: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Rotation of the detector about the x axis (roll)

yaw: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Rotation of the detector about the y axis (yaw)

beam_center_x: (optional) *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) *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) *NX_FLOAT* {units=*NX_LENGTH*} <=

Size of each detector pixel. If it is scalar all pixels are the same size

y_pixel_size: (optional) *NX_FLOAT* {units=*NX_LENGTH*} <=

Size of each detector pixel. If it is scalar all pixels are the same size

SOURCE: (optional) *NXsource* <=

Description of the radiation source.

@canSAS_class: (required) *NX_CHAR*

Official canSAS group: NXcanSAS (applications); SASsource

Obligatory value: `SASsource`

radiation: (optional) *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 (λ) 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`

name: (optional) *NX_CHAR*

Optional name for this data processing or analysis step

date: (optional) *NX_DATE_TIME* *<=*

Optional date for this data processing or analysis step.Page 336, 2

description: (optional) *NX_CHAR*

Optional description for this data processing or analysis step

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 *SAStransmission_spectrum* element

This describes certain data obtained from a variable-wavelength source such as pulsed-neutron source.

The name of each *SAStransmission_spectrum* group must be unique within a SASentry group. Suggest using names such as `sastransmission_spectrum01`. **@canSAS_class**: (required) *NX_CHAR*

Official canSAS group: NXcanSAS (applications); SAStransmission_spectrum

Obligatory value: `SAStransmission_spectrum`

@signal: (required) *NX_CHAR <=*

Name of the default data field.

Obligatory value:

- `T`: For **SAStransmission_spectrum**, this is always "T".

@T_axes: (required) *NX_CHAR*

Obligatory value:

- `T`: the wavelengths field (as axis coordinates) corresponding to this transmission

@name: (required) *NX_CHAR*

Identify what type of spectrum is being described. It is expected that this value will take either of these two values:

value	meaning
sample	measurement with the sample and container
can	measurement with just the container

@timestamp: (optional) *NX_DATE_TIME*

ISO-8601 time[Page 336, 2]

lambda: (required) *NX_NUMBER* {units=*NX_WAVELENGTH*}

Wavelength of the radiation.

This array is of the same shape as `T` and `Tdev`.

T: (required) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

Transmission values (I/I_0) as a function of wavelength.

This array is of the same shape as `lambda` and `Tdev`. **@uncertainties**: (required) *NX_CHAR*

Names the dataset (in this SASdata group) that provides the uncertainty of each transmission T to be used for data analysis. The name of the dataset containing the T uncertainty is expected to be `Tdev`.

Typically:

@uncertainties="Tdev"

Tdev: (required) *NX_NUMBER* {units=*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*
- */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*
- */NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM@timestamp-attribute*
- */NXcanSAS/ENTRY@canSAS_class-attribute*
- */NXcanSAS/ENTRY@default-attribute*
- */NXcanSAS/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXcanSAS.nxdl.xml

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 definitly 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.

- */NXdirecttof/entry-group*
- */NXdirecttof/entry/definition-field*
- */NXdirecttof/entry/INSTRUMENT-group*
- */NXdirecttof/entry/INSTRUMENT/disk_chopper-group*

- */NXdirecttof/entry/INSTRUMENT/disk_chopper/energy-field*
- */NXdirecttof/entry/INSTRUMENT/disk_chopper/rotation_speed-field*
- */NXdirecttof/entry/INSTRUMENT/fermi_chopper-group*
- */NXdirecttof/entry/INSTRUMENT/fermi_chopper/energy-field*
- */NXdirecttof/entry/INSTRUMENT/fermi_chopper/rotation_speed-field*
- */NXdirecttof/entry/start_time-field*
- */NXdirecttof/entry/title-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXdirecttof.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) *NX_CHAR* <=

name: (required) *NX_CHAR* <=

probe: (required) *NX_CHAR* <=

Obligatory value: `x-ray`

monochromator: (required) *NXmonochromator* <=

wavelength: (required) *NX_FLOAT* <=

fluorescence: (required) *NXdetector* <=

data: (required) *NX_INT* (Rank: 1, Dimensions: [nE])

energy: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nE])

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_INT*

data: (required) *NXdata <=*

energy: *link* (suggested target: `/entry/instrument/fluorescence/energy`)

data: *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: `NXiqproc`

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

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 thie 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*
- */NXlauetof/entry/instrument/detector/distance-field*
- */NXlauetof/entry/instrument/detector/polar_angle-field*
- */NXlauetof/entry/instrument/detector/time_of_flight-field*
- */NXlauetof/entry/instrument/detector/x_pixel_size-field*
- */NXlauetof/entry/instrument/detector/y_pixel_size-field*
- */NXlauetof/entry/name-group*
- */NXlauetof/entry/name/data-link*

- */NXlauetof/entry/name/time_of_flight-link*
- */NXlauetof/entry/sample-group*
- */NXlauetof/entry/sample/name-field*
- */NXlauetof/entry/sample/orientation_matrix-field*
- */NXlauetof/entry/sample/unit_cell-field*

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:

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: `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_channel*, *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 exposures.

INSTRUMENT: (required) *NXinstrument <=*

name: (required) *NX_CHAR <=*

Name of instrument. Consistency with the controlled vocabulary beamline 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_GROUP: (recommended) *NXdetector_group <=*

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:

```
DTL         DTR
DLL         DLR
```

We could have:

```
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 dectector 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.

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: (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_NUMBER* <=

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_NUMBER* <=

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 definiton 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) *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) *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) *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) *NX_NUMBER* {units=*NX_LENGTH*} <=

Offset of the module in regards to the origin of the detector in an arbitrary direction. **@transformation_type**: (required) *NX_CHAR* <=

Obligatory value: `translation`

@vector: (required) *NX_NUMBER* <=

@offset: (required) *NX_NUMBER* <=

@depends_on: (required) *NX_CHAR* <=

fast_pixel_direction: (required) *NX_NUMBER* {units=*NX_LENGTH*} <=

Values along the direction of *fastest varying* pixel direction. The direction itself is given through the vector attribute. **@transformation_type**: (required) *NX_CHAR* <=

Obligatory value: `translation`

@vector: (required) *NX_NUMBER* <=

@offset: (required) *NX_NUMBER* <=

@depends_on: (required) *NX_CHAR* <=

slow_pixel_direction: (required) *NX_NUMBER* {units=*NX_LENGTH*} <=

Values along the direction of *slowest varying* pixel direction. The direction itself is given through the vector attribute. **@transformation_type**: (required) *NX_CHAR* <=

Obligatory value: `translation`

@vector: (required) *NX_NUMBER* <=

@offset: (required) *NX_NUMBER* <=

@depends_on: (required) *NX_CHAR* <=

CHANNELNAME_channel: (required) *NXdetector_channel* <=

Group containing the description and metadata for a single channel from a multi-channel detector.

Given an *NXdata* group linked as part of an NXdetector group that has an axis with named channels (see the example in *NXdata*), the NXdetector will have a series of NXdetector_channel groups, one for each channel, named CHANNELNAME_channel.

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 monchromatic 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 **np** by **m** (slow to fast) of the relative weights of the corresponding wavelengths in `incident_wavelength`.

incident_wavelength_spread: (optional) *NX_FLOAT* {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.

flux: (optional) *NX_FLOAT* {units=*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.

total_flux: (optional) *NX_FLOAT* {units=*NX_FREQUENCY*}

Flux incident on beam plane in photons per second. In other words this is the *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.

flux_integrated: (optional) *NX_FLOAT* {units=*NX_PER_AREA*}

Flux density incident on beam plane area in photons per unit area. In other words this is the *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.

total_flux_integrated: (optional) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Flux incident on beam plane in photons. In other words this is the *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.

incident_beam_size: (recommended) *NX_FLOAT* (Rank: 1, Dimensions: [2]) {units=*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

profile: (recommended) *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

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* <=

This group is intended for use cases that do not fit the *incident_wavelength* and *incident_wavelength_weights* fields above, perhaps for example a 2D spectrometer.

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 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/CHANNELNAME_channel-group*
- */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/countrate_correction_lookup_table-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*
- */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/module_offset@vector-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
- /NXmx/ENTRY/INSTRUMENT/time_zone-field
- /NXmx/ENTRY/SAMPLE-group
- /NXmx/ENTRY/SAMPLE/depends_on-field
- /NXmx/ENTRY/SAMPLE/name-field
- /NXmx/ENTRY/SAMPLE/temperature-field
- /NXmx/ENTRY/SAMPLE/TRANSFORMATIONS-group
- /NXmx/ENTRY/SOURCE-group
- /NXmx/ENTRY/SOURCE/name-field
- /NXmx/ENTRY/SOURCE/name@short_name-attribute
- /NXmx/ENTRY/start_time-field
- /NXmx/ENTRY/title-field
- /NXmx/ENTRY@version-attribute

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXmx.nxdl.xml

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 NXDL schema to which this file conforms

Obligatory value: `NXrefscan`

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* <=

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

NXreftof

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*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXreftof.nxdl.xml

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 (λ) 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) *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.

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) *NXsample* *<=*

name: (required) *NX_CHAR* *<=*

Descriptive name of sample.

aequatorial_angle: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

MONITOR: (optional) *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* *<=*

@signal: (optional) *NX_CHAR* *<=*

Name the *plottable* field. The link here defines this name as `data`.

Obligatory value: `data`

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.

- */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/COLLIMATOR/GEOMETRY/SHAPE/shape-field*
- */NXsas/ENTRY/INSTRUMENT/COLLIMATOR/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/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.

nXPixel: nXPixel description

nYPixel: nYPixel 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

aequatorial_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*
- */NXsastof/ENTRY/instrument/detector/time_of_flight-field*
- */NXsastof/ENTRY/instrument/detector/x_pixel_size-field*
- */NXsastof/ENTRY/instrument/detector/y_pixel_size-field*
- */NXsastof/ENTRY/instrument/name-field*
- */NXsastof/ENTRY/instrument/source-group*
- */NXsastof/ENTRY/instrument/source/name-field*
- */NXsastof/ENTRY/instrument/source/probe-field*
- */NXsastof/ENTRY/instrument/source/type-field*
- */NXsastof/ENTRY/sample-group*
- */NXsastof/ENTRY/sample/aequatorial_angle-field*
- */NXsastof/ENTRY/sample/name-field*
- */NXsastof/ENTRY/start_time-field*
- */NXsastof/ENTRY/title-field*
- */NXsastof/ENTRY@entry-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXsastof.nxdl.xml

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/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*
- */NXspe/ENTRY/SAMPLE-group*
- */NXspe/ENTRY/SAMPLE/rotation_angle-field*
- */NXspe/ENTRY/SAMPLE/seblock-field*
- */NXspe/ENTRY/SAMPLE/temperature-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_DIMENSIONLESS*}

qk: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_DIMENSIONLESS*}

ql: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_DIMENSIONLESS*}

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`)

qh: *link* (suggested target: `/NXentry/NXsample/qh`)

qk: *link* (suggested target: `/NXentry/NXsample/qk`)

ql: *link* (suggested target: `/NXentry/NXsample/ql`)

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.

- */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*} <=

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*} <=

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) *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`)

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.

- */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*

- */NXtofsingle/entry/pre_sample_flightpath-field*
- */NXtofsingle/entry/SAMPLE-group*
- */NXtofsingle/entry/SAMPLE/name-field*
- */NXtofsingle/entry/SAMPLE/nature-field*
- */NXtofsingle/entry/start_time-field*
- */NXtofsingle/entry/title-field*
- */NXtofsingle/entry/user-group*
- */NXtofsingle/entry/user/name-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`

detector: (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* <=

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*

- */NXtomo/entry/start_time-field*
- */NXtomo/entry/title-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXtomo.nxdl.xml

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*

NXDL 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/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.

- */NXxbase/entry-group*
- */NXxbase/entry/control-group*
- */NXxbase/entry/control/integral-field*
- */NXxbase/entry/control/mode-field*
- */NXxbase/entry/control/preset-field*
- */NXxbase/entry/DATA-group*
- */NXxbase/entry/DATA/data-link*
- */NXxbase/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*

- */NXxeuler/entry/name/phi-link*
- */NXxeuler/entry/name/polar_angle-link*
- */NXxeuler/entry/name/rotation_angle-link*
- */NXxeuler/entry/sample-group*
- */NXxeuler/entry/sample/chi-field*
- */NXxeuler/entry/sample/phi-field*
- */NXxeuler/entry/sample/rotation_angle-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxeuler.nxdl.xml

NXxkappa

Status:

application definition, extends *NXxbase*

Description:

raw data from a kappa geometry (CAD4) single crystal diffractometer, extends *NXxbase*

This is the application definition for raw data from a kappa geometry (CAD4) single crystal diffractometer. It extends *NXxbase*, so the full definition is the content of *NXxbase* plus the data defined here.

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: `NXxkappa`

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.

- */NXxkappa/entry-group*
- */NXxkappa/entry/definition-field*
- */NXxkappa/entry/instrument-group*
- */NXxkappa/entry/instrument/detector-group*
- */NXxkappa/entry/instrument/detector/polar_angle-field*
- */NXxkappa/entry/name-group*
- */NXxkappa/entry/name/kappa-link*
- */NXxkappa/entry/name/phi-link*
- */NXxkappa/entry/name/polar_angle-link*
- */NXxkappa/entry/name/rotation_angle-link*
- */NXxkappa/entry/sample-group*
- */NXxkappa/entry/sample/alpha-field*
- */NXxkappa/entry/sample/kappa-field*
- */NXxkappa/entry/sample/phi-field*
- */NXxkappa/entry/sample/rotation_angle-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxkappa.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

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

NXxnb

Status:

application definition, extends *NXxbase*

Description:

raw data from a single crystal diffractometer, extends *NXxbase*

This is the application definition for raw data from a single crystal diffractometer measuring in normal beam mode. It extends *NXxbase*, so the full definition is the content of *NXxbase* plus the data defined here. All angles are logged in order to support 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: `NXxnb`

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 NXDL 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-field*
- */NXxrot/entry/sample/rotation_angle_step-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxrot.nxdl.xml

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.

Some contributions are grouped together:

Optical Spectroscopy

Multi-dimensional Photoemission Spectroscopy

Atom Probe Microscopy

Electron Microscopy

Transport Measurements

Geometry and Microstructures

and others are simply listed here:

NXaberration
Quantified aberration coefficient in an aberration_model.

NXaberration_model
Models for aberrations of electro-magnetic lenses in electron microscopy.

NXaberration_model_ceos
CEOS definitions/model for aberrations of electro-magnetic lenses.

NXaberration_model_nion
NION definitions/model for aberrations of electro-magnetic lenses.

NXadc
Analog-to-digital converter component/integrated circuit.

NXaperture_em
Details of an individual aperture for beams in electron microscopy.

NXbeam_path
A beam path consisting of one or more optical elements.

NXbeam_splitter
A beam splitter, i.e. a device splitting the light into two or more beams.

NXcalibration
Subclass of NXprocess to describe post-processing calibrations.

NXchamber
Component of an instrument to store or place objects and specimens.

NXchemical_composition
(Chemical) composition of a sample or a set of things.

NXcircuit_board
Circuit board with e.g. ADC and/or DAC electronic components.

NXclustering
: Metadata to the results of a clustering analysis.

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.

NXcs_computer
: Computer science description of a set of computing nodes.

NXcs_cpu
: Computer science description of a central processing unit (CPU) of a computer.

NXcs_filter_boolean_mask
: Computer science base class for packing and unpacking booleans.

NXcs_gpu
: Computer science description of a graphic processing unit (GPU) of a computer.

NXcs_io_obj
: Computer science description of a storage object in an input/output system.

NXcs_io_sys
: Computer science description of system of a computer.

NXcs_mm_sys
: Computer science description of a main memory system of a computer.

NXcs_prng
: Computer science description of pseudo-random number generator.

NXcs_profiling
: Computer science description for summary performance/profiling data of an application.

NXcs_profiling_event
: Computer science description of a profiling event.

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.

NXdac
: Digital-to-analog converter component/integrated circuit.

NXdeflector
: Deflectors as they are used e.g. in an electron analyser.

NXdelocalization
: Base class to describe the delocalization of point-like objects on a grid.

NXdispersion
: A dispersion denoting a sum of different dispersions.

NXdispersion_function
This describes a dispersion function for a material or layer

NXdispersion_repeated_parameter
A repeated parameter for a dispersion function

NXdispersion_single_parameter
A single parameter for a dispersion function

NXdispersion_table
A dispersion table denoting energy, dielectric function tabulated values.

NXdispersive_material
NXdispersion

NXdistortion
Subclass of NXprocess to describe post-processing distortion correction.

NXebeam_column
Container for components to form a controlled beam in electron microscopy.

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 of a sample during a session on an electron microscope.

NXem_ebsd
Application definition for collecting and indexing Kikuchi pattern into orientation maps.

NXem_ebsd_conventions
Conventions for rotations and coordinate systems to interpret EBSD data.

NXem_ebsd_crystal_structure_model
Crystal structure phase models used for indexing Kikuchi pattern.

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.

NXfabrication
Details about a component as defined by its manufacturer.

NXfiber
An optical fiber, e.g. glass fiber.

NXgraph_edge_set
A set of (eventually directed) edges which connect nodes/vertices of a graph.

NXgraph_node_set
A set of nodes/vertices in space representing members of a graph.

NXgraph_root
An instance of a graph.

NXibeam_column
Container for components of a focused-ion-beam (FIB) system.

NXimage_set
Container for reporting a set of images taken.

NXimage_set_em_adf
Container for reporting a set of annular dark field images.

NXimage_set_em_kikuchi
Measured set of electron backscatter diffraction data, aka Kikuchi pattern.

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.

NXisocontour
Computational geometry description of isocontouring/phase-fields in Euclidean space.

NXiv_temp
Application definition for temperature-dependent IV curve measurements.

NXlab_electro_chemo_mechanical_preparation
Grinding and polishing of a sample using abrasives in a wet lab.

NXlab_sample_mounting
Embedding of a sample in a medium for easing processability.

NXlens_em
Description of an electro-magnetic lens or a compound lens.

NXlens_opt
Description of an optical lens.

NXmagnetic_kicker
definition for a magnetic kicker.

NXmanipulator
Extension of NXpositioner to include fields to describe the use of manipulators

NXmatch_filter
Settings of a filter to select or remove entries based on their value.

NXmpes
This is the most general application definition for multidimensional

NXopt
An application definition for optical spectroscopy experiments.

NXoptical_system_em
A container for qualifying an electron optical system.

NXorientation_set
Details about individual orientations of a set of objects.

NXpeak
Description of peaks, their functional form or measured support.

NXpid
Contains the settings of a PID controller.

NXpolarizer_opt
An optical polarizer.

NXprogram
Base class to describe a software tool or library.

NXpulser_apm
Metadata for laser- and/or voltage-pulsing in atom probe microscopy.

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.

NXsensor_scan
Application definition for a generic scan using sensors.

NXseparator
definition for an electrostatic separator.

NXsimilarity_grouping
Metadata to the results of a similarity grouping analysis.

NXslip_system_set
Base class for describing a set of crystallographic slip systems.

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

NXspatial_filter
Spatial filter to filter entries within a region-of-interest based on their

NXspectrum_set
Container for reporting a set of spectra.

NXspectrum_set_em_eels
Container for reporting a set of electron energy loss (EELS) 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.

NXsubsampling_filter
Settings of a filter to sample entries based on their value.

NXtransmission
Application definition for transmission experiments

NXwaveplate
A waveplate or retarder.

NXxpcs
X-ray Photon Correlation Spectroscopy (XPCS) data (results).

Electron microscopy

Introduction

Partner consortia in the German National Research Data Infrastructure are here e.g. NFDI-MatWerk, NFDI4Ing, NFDI-BioImage, NFDI-Microbiota, NFDI4Health, and e.g. NFDI-Neuro.

Electron microscopes are functionally very customizable tools: Examples include multi-signal/-modal analyses which are frequently realized as on-the-fly computational analyses, regularly switching between GUI-based instrument control, computational steps, and more and more using high-throughput stream-based processing. Also artificial intelligence methods are increasingly used and are becoming more closely interconnected with classical modes of controlling the instrument and perform data processing. A challenge in electron microscopy is that these steps are often executed within commercial integrated control and analysis software. This makes it difficult to keep track of workflows in a technology-partner-agnostic, i.e. interdisciplinary manner.

Application Definitions

We acknowledge that it can be difficult to agree on a single application definition which is generally enough applicable yet not unnecessarily complex and useful for applications across a variety of instruments, technology partners, and instrument use cases. In what follows, the proposal conceptualizes first the basic components of an electron microscope and the usual workflow of how an electron microscope is used for collecting data with detectors via probing radiation-specimen-matter interaction mechanisms.

In summary, scientists place a specimen/sample into the microscope, calibrate the instrument, take measurements, and may perform experiments, prepare their specimens with a focused ion beam, calibrate again, and take other measurements, before their session on the instrument ends. In between virtually all of these steps data are collected and stream in from different detectors probing different physical mechanisms of the interaction between electrons or other types of radiation with the specimen.

A microscope session ends with the scientist removing the specimen from the instrument or parking it so that the next user can start a session. Occasionally, service technicians perform calibrations and maintenance which also can be described as a session on the microscope. We have provided base classes to describe these steps and events and an application definition for electron microscopy.

***NXem*:**
A general application definition which explores the possibilities of electron microscopes.

Base Classes

The following base classes are proposed to support modularizing the storage of pieces of information:

***NXaberration_model*, *NXaberration_model_ceos*, *NXaberration_model_nion*, *NXaberration*:**
Base classes to describe procedures and values for the calibration of aberrations based on either CEOS or Nion.

***NXaperture_em*:**
A base class to describe an aperture.

***NXchamber*:**
A base class to describe the chamber as a part of the microscope or storage unit for transferring specimens in between or within an instrument.

***NXcoordinate_system_set*:**
A base class to describe different coordinate systems used and/or to be harmonized or transformed into one another when interpreting the dataset.

***NXcorrector_cs*:**
A base class to describe details about corrective lens or compound lens devices which reduce the aberration of an electron beam.

***NXebeam_column*:**
A base class serving the possibility to group the components relevant for generating and shaping the electron beam.

***NXevent_data_em*:**
A base class representing a container to hold time-stamped and microscope-state- annotated data during a session at an electron microscope.

***NXevent_data_em_set*:**
A base class to group all *NXevent_data_em* instances.

***NXibeam_column*:**
A base class serving the possibility to group the components relevant for generating and shaping an ion beam of an instrument to offer focused-ion beam (milling) capabilities.

***NXimage_set*:**
Base classes for storing acquisition details for individual images or stacks of images. Specialized versions can be defined and use controlled vocabulary terms for group name prefixes like **adf** annular dark field, **bf** bright field, **bse** backscattered electron, **chamber** camera to monitor the stage and chamber, **df** darkfield, **diffrac** diffraction, **ecci** electron channeling contrast imaging, **kikuchi** electron backscatter diffraction, **ronchigram** - convergent beam diffraction pattern, or **se** secondary electron.

***NXinteraction_vol_em*:**
A base class to describe details about e.g. the simulated or known volume of interaction of the electrons with the specimen.

NXion:
A base class to describe charged molecular ions with an adjustable number of atoms/isotopes building each ion. Right now the maximum number of atoms supported building a molecular ion is 32. Suggestions made in reference DOI: 10.1017/S1431927621012241 are used to map isotope to hash values with which all possible isotopes can be described.

NXlens_em:
A base class to detail an electro-magnetic lens. In practice, an electron microscope has many such lenses. It is possible to specify as many lenses as necessary to represent eventually each single lens of the microscope and thus describe how the lenses are affecting the electron beam. This can offer opportunities for developers of software tools which strive to model the instrument e.g. to create digital twins of the instrument. We understand there is still a way to go with this to arrive there though. Consequently, we suggest to focus first on which details should be collected for a lens as a component so that developers of application definitions can take immediate advantage of this work.

NXfabrication:
A base class to bundle manufacturer/technology-partner-specific details about a component or device of an instrument.

NXoptical_system_em:
A base class to store for now qualitative and quantitative values of frequent interest which are affected by the interplay of the components and state of an electron microscope. Examples are the semiconvergence angle or the depth of field and depth of focus, the magnification, or the camera length.

NXpeak:
A base class to describe peaks mathematically so that it can be used to detail how peaks in mass-to-charge-state ratio histograms (aka mass spectra) are defined and labelled as iontypes.

NXpump:
A base class to describe details about a pump in an instrument.

NXscanbox_em:
A base class to represent the component of an electron microscope which realizes a controlled deflection (and eventually shift, blanking, and/or descanning) of the electron beam to illuminate the specimen in a controlled manner. This can be used to document the scan pattern.

NXspectrum_set:
Base class and specializations comparable to NXimage_set but for storing spectra. Specialized base classes should use controlled vocabulary items as prefixes such as **eels** electron energy loss spectroscopy, **xray** X-ray spectroscopy (EDS/STEM, EDX, SEM/EDX, SEM/EDS), **auger** Auger spectroscopy, or **cathodolum** for cathodoluminescence spectra.

NXstage_lab:
As it was mentioned for atom probe microscopy, this is a base class to describe the stage/specimen holder which offers place for the documentation of the small-scale laboratory functionalities which modern stages of electron microscopes frequently offer.

Method-specific concepts and their usage in application definitions

It became clear during the design of the electron-microscopy-specific additions to NeXus that there are sets of pieces of information (data and metadata) which are relevant for a given experiment but have usually only few connections to the detailed description of the workflow of processing these data into knowledge, motivating the granularization of these pieces of information in their own application definition. In fact, one limitation of application definitions in NeXus is that they define a set of constraints on their graph of controlled concepts and terms. If we take for example diffraction experiments with an electron microscope it is usually the case that (diffraction) patterns are collected in the session at the microscope but all scientifically relevant conclusions are drawn later, i.e. through post-processing these data. These numerical and algorithmic steps define computational workflows where data from the application definitions such as NXem are used as input but many additional concepts and constraints may apply without any need for changing constraints on fields or groups of NXem. If we were to modify NXem for these cases, NXem would likely combinatorially diverge as every different combination of required constraints would demand having their own but almost similar application definition. For this reason, we propose to define the following base classes:

More consolidation through the use of NXsubentry classes should be considered in the future.

***NXem_ebsd*:**
Application definition for collecting and indexing Kikuchi pattern into orientation maps for the two-dimensional, three- and four-dimensional case.

Several new base classes are used by this application definition.

***NXem_ebsd_conventions*:**
A collection of reference frames and rotation conventions necessary to interpret the alignment and orientation data.

***NXem_ebsd_crystal_structure_model*:**
A description of a crystalline phase/structure used for a forward simulation using kinetic or dynamic diffraction theory to generate simulated diffraction pattern against which measured pattern can be indexed.

Optical Spectroscopy

Ellipsometry

Ellipsometry is an optical characterization method to describe optical properties of interfaces and thickness of films. The measurements are based on determining how the polarization state of light changes upon transmission and reflection. Interpretation is based on Fresnel equations and numerical models of the optical properties of the materials.

In the application definition we provide a minimum set of description elements allowing for a reproducible recording of ellipsometry measurements.

Application Definitions

***NXopt*:**
A generic application definition for optical spectroscopy measurements, including complex systems up to variable angle spectroscopic ellipsometry.

***NXellipsometry*:**
An application definition for ellipsometry measurements, including complex systems up to variable angle spectroscopic ellipsometry.

Base Classes

This is the set of base classes for describing an optical experiment.

NXbeam_path
: A beam path consisting of one or more optical elements.

  NXbeam_path is used in NXopt to describe the beam path, i.e. the arrangement of optical elements between the excitation source and the sample, or between the sample and the detector unit.

NXbeam_splitter
: A beam splitter, i.e. a device splitting the light into two or more beams.

  Use two or more NXbeam_paths to describe the beam paths after the beam splitter. In the dependency chain of the new beam paths, the first elements each point to this beam splitter, as this is the previous element.

NXfiber
: An optical fiber, e.g. glass fiber.

NXlens_opt
: Description of an optical lens.

NXpolarizer_opt
: An optical polarizer.

NXwaveplate
: A waveplate or retarder.

NXenvironment
: Specify external parameters that have influenced the sample, such as the surrounding medium, and varied parameters e.g. temperature, pressure, pH value, optical excitation etc.

Dispersive Material

A dispersive material is a description for the optical dispersion of materials. This description may be used to store optical model data from an ellipsometric analysis (or any other technique) or to build a database of optical constants for optical properties of materials.

Application Definition

NXdispersive_material:
: An application definition to describe the dispersive properties of a material. The material may be isotropic, uniaxial or biaxial. Hence, it may contain up to three dispersive functions or tables.

Base Classes

There is a set of base classes for describing a dispersion.

NXdispersion
: This is an umbrella base class for a group of dispersion functions to describe the material. For a simple dispersion it may contain only on NXdispersion_function or NXdispersion_table entry. If it contains multiple entries the actual dispersion is the sum of all dispersion functions and tables. This allows for, e.g. splitting real and imaginary parts and describing them seperately or adding a dielectric background (e.g. Sellmeier model) to an oscillator model (e.g. Lorentz).

NXdispersion_function
This dispersion is described by a function and its single and repeated parameter values. It follows a formula of the form `eps = eps_inf + sum[A * lambda ** 2 / (lambda ** 2 - B ** 2)]` (Sellmeier formula). See the formula grammar below for an ebnf grammar for this form.

NXdispersion_single_parameter
This denotes a parameter which is used outside the summed part of a dispersion function, e.g. `eps_inf` in the formula example above.

NXdispersion_repeated_parameter
This denotes arrays of repeated parameters which are used to build a sum of parameter values, e.g. `A` and `B` are repeated parameters in the formula above.

NXdispersion_table
This describes a tabular dispersion where the permittivity is an array versus wavelength or energy.

Formula Grammar

Below you find a grammar to which the formula should adhere and which can be used to parse and evaluate the dispersion function. The terms `single_param_name` and `param_name` should be filled with the respective single and repeated params from the stored data. The grammer is written in the EBNF dialect of Lark, which is a parsing toolkit for python. It is easily translatable to general EBNF and other parser generator dialects. Here is a reference implementation in Rust/Python with a grammar written in lalrpop.

```
?assignment: "eps" "=" kkr_expression -> eps
          | "n" "=" kkr_expression -> n

?kkr_expression: expression
               | "<kkr>" "+" "1j" "*" term -> kkr_term

?expression: term
            | expression "+" term -> add
            | expression "-" term -> sub

?term: factor
      | term "*" factor -> mul
      | term "/" factor -> div

?factor: power
      | power "**" power -> power


?power: "(" expression ")"
      | FUNC "(" expression ")" -> func
      | "sum" "[" repeated_expression "]" -> sum_expr
      | NAME -> single_param_name
      | SIGNED_NUMBER -> number
      | BUILTIN -> builtin

?repeated_expression: repeated_term
                   | repeated_expression "+" repeated_term -> add
                   | repeated_expression "-" repeated_term -> sub
```

(continues on next page)

(continued from previous page)

```
?repeated_term: repeated_factor
              | repeated_term "*" repeated_factor -> mul
              | repeated_term "/" repeated_factor -> div

?repeated_factor: repeated_power
                | repeated_power "**" repeated_power -> power

?repeated_power: "(" repeated_expression ")"
               | FUNC "(" repeated_expression ")" -> func
               | SIGNED_NUMBER -> number
               | NAME -> param_name
               | BUILTIN -> builtin

FUNC.1: "sin" | "cos" | "tan" | "sqrt" | "dawsn" | "ln" | "log" | "heaviside"
BUILTIN.1: "1j" | "pi" | "eps_0" | "hbar" | "h" | "c"

%import common.CNAME -> NAME
%import common.SIGNED_NUMBER
%import common.WS_INLINE

%ignore WS_INLINE
```

Photoemission & core-level spectroscopy

Introduction

Set of data storage objects to describe photoemission experiments including x-ray photoelectron spectroscopy (XPS), ultraviolet photoelectron spectroscopy (UPS), hard x-ray photoelectron spectroscopy (HAXPES), angle-resolved photoemission spectroscopy (ARPES), two-photon photoemission (2PPE) and photoemission electron microscopy (PEEM). Also includes descriptors for advanced specializations, such as spin-resolution, time resolution, near-ambient pressure conditions, dichroism etc.

Application Definitions

NXmpes:
A general appdef with minimalistic metadata requirements, apt to describe all photemission experiments.

Base Classes

NXelectronanalyser:
A base class to describe electron kinetic energy analizers. Contains the collective characteristics of the instrument such as energy resolution, and includes the following subclasses:

NXcollectioncolumn:
Base class to describe the set of electronic lenses in the electron collection column (standard, PEEM, momentum-microscope, etc.).

NXenergydispersion:
Base class to describe the energy dispersion sytem (hemispherical, time-of-flight, etc.).

NXspindispersion:
Base class to describe the set of electronic lenses in the electron collection column.

NXmanipulator:
A base class to describe the complex manipulators used in photoemission experiments, often with > 4 degrees of freedom, cryogenic cooling and other advanced features.

Three base classes to describe data processing, which can be used as subclasses of *NXprocess* if describing post-processing or as subclasses of *NXdetector* if describing live, electronics level processing:

NXcalibration:
A base class to describe the 1D calibration of an axis, with a function mapping a raw data scale to a calibrated scale with the same number of points.

NXdistortion:
A base class to describe the 2D distortion correction of an axis, with a matrix mapping a raw data image to a undistorted image.

NXregistration:
A base class to describe the rigid transformations that are applied to an image. May be redundant as they can be described with *NXtransformations*.

Common Base Classes

There are two related base classes that are common to other techniques:

NXlens_em:
A class to describe all types of lenses. Includes electrostatic lenses for electron energy analysers.

NXdeflector
A class to describe all kinds of deflectors, including electrostatic and magnetostatic deflectors for electron energy analysers.

Atom-probe tomography

Introduction

Set of data storage objects to describe the acquisition/measurement side, the reconstruction, and the ranging for atom probe microscopy experiments. The data storage objects can be useful as well for field-ion microscopy experiments.

Application Definition

It is proposed to use one application definition to serve atom probe tomography and field-ion microscopy measurements, i.e. the data collection with the instrument:

NXapm:
A general application definition with many detailed places for leaving metadata and computational steps described which are commonly used when reporting the measurement of atom probe data including also detector hit data, as well as how to proceed with reconstructing atom positions from these data, and how to store details about definitions made, which describe how mass-to-charge-state ratio values are mapped to iontypes in a process called ranging.

Base Classes

The following base classes are proposed to support modularizing the storage of pieces of information:

***NXchamber*:**
A base class to describe a component of the instrument which houses other components. A chamber may offer a controlled atmosphere to execute an experiment and/or offer functionalities for storing and loading specimens.

NXcoordinate_system_set
A base class to describe different coordinate systems used and/or to be harmonized or transformed into one another when interpreting the dataset.

***NXion*:**
A base class to describe charged molecular ions with an adjustable number of atoms/isotopes building each ion. Right now the maximum number of atoms supported building a molecular ion is 32. Suggestions made in reference DOI: 10.1017/S1431927621012241 are used to map isotope to hash values with which all possible isotopes can be described.

***NXfabrication*:**
A base class to bundle manufacturer/technology-partner-specific details about a component or device of an instrument.

***NXpeak*:**
A base class to describe peaks mathematically to detail how peaks in mass-to-charge-state ratio histograms (aka mass spectra) are defined and labelled as iontypes.

***NXpump*:**
A base class to describe details about pump(s) of an instrument.

***NXpulser_apm*:**
A base class to describe the high-voltage and/or laser pulsing capabilities of an atom probe microscope.

***NXreflectron*:**
A base class to describe a kinetic-energy-sensitive filtering device for time of flight (ToF) mass spectrometry.

***NXstage_lab*:**
A base class to describe the specimen fixture including the cryo-head. Nowadays, these stages represent small-scale laboratory platforms. Therefore, there is a need to define the characteristics of such stages in more detail, especially in light of in-situ experiments. Many similarities exists between a stage in an electron microscope and one in an atom probe instrument. Both offer fixture functionalities and additional components for applying e.g. stimuli on the specimen.

Microscopy experiments, not only taking into account those performed on commercial instruments, offer the user usually to apply a set of data processing steps. Some of them are frequently applied on-the-fly. For now we represent these steps with specifically named instances of the *NXprocess* base class.

Like every research community data processing steps are essential to transform measurements into knowledge. These processing steps should be documented to enable reproducible research and learn how pieces of information were connected. In what follows, an example is presented how an open-source community software can be modified to use descriptions of these computational steps. The respective research software here is the paraprobe-toolbox

apmtools

One tool is the paraprobe-toolbox software package in the the apmtools container. The software is developed by M. Kühbach et al..

Here we show how NeXus is used to consistently define application definitions for scientific software in the field of atom probe. In this community the paraprobe-toolbox is an example of an open-source parallelized software for analyzing point cloud data, for assessing meshes in 3D continuum space, and for studying the effects of parameterization on descriptors which describe the micro- and nanostructure of materials that are studied with atom probe microscopy.

The need for a thorough documentation of the tools in not only the paraprobe-toolbox was motivated by several needs:

First, users of software would like to better understand and also be able to study for themselves which individual parameters and settings for each tool exist and how configuring these affects their analyses quantitatively.

Second, scientific software like the tools in the paraprobe-toolbox implement a numerical/algorithmical (computational) workflow whereby data from multiple input sources (like previous analysis results) are processed and carried through more involved analysis within several steps inside the tool. The tool then creates output as files.

Individual tools of paraprobe-toolbox are developed in C/C++ and/or Python. Provenance tracking is useful as it is one component and requirement for making workflows exactly numerically reproducible and thereby empower scientists to fulfill better the "R", i.e. reproducibility of their daily FAIR research practices.

The paraprobe-toolbox is one example of a software which implements a workflow via a sequence of operations executed within a jupyter notebook (or Python script respectively). Specifically, individual tools are chained. Convenience functions are available to create well-defined input/configuration files for each tool. These config files instruct the tool upon processing.

In this design, each workflow step (with a tool) is in fact a pair (or triple) of at least two sub-steps: i) the creation of a configuration file, ii) the actual analysis using the Python/or C/C++ tools, iii) the optional post-processing/visualizing of the results inside the NeXus/HDF5 files generated from each tool run using other software.

What has been achieved so far?

This proposal summarizes work of members of the FAIRmat project, which is part of the German National Research Data Infrastructure aimed at a change of the paraprobe-toolbox and its interaction with files for all tools so that only well-defined configuration files are accepted as input and results end up as specifically formatted output. For this NeXus application definitions are used.

Data and metadata between the tools are exchanged with NeXus/HDF5 files. Specifically, we created for each tool an application definition (see below) which details all possible settings and options for the configuration as to guide users. The config(uration) files are currently implemented as HDF5 files, whose content matches to the naming conventions of the respective *config* application definition for each tool. As an example NXapm_paraprobe_config_surfacer specifies how a configuration file for the paraprobe-surfacer tool should be formatted and which parameter it should and/or may contain.

That is each config file uses a controlled vocabulary of terms. Furthermore, the config files store a SHA256 checksum for each input file. This implements a full provenance tracking on the input files along the processing chain/workflow.

As an example, a user may first range their reconstruction and then compute correlation functions. The config file for the ranging tool stores the files which hold the reconstructed ion position and ranging definitions. The ranging tool generates a results file with the ion type labels stored. This results file is formatted according to the tool-specific *results* application definition. This results file and the reconstruction is imported by the spatial statistics tool which again keeps track of all files.

This design makes it possible to rigorously trace which numerical results were achieved with a specific input and settings using specifically-versioned tools.

We understand that this additional handling of metadata and provenance tracking may not be at first glance super relevant for scientists or appears to be an unnecessarily complex feature. There is indeed an additional layer of work which came with the development and maintenance of this functionality.

However, we are convinced that this is the preferred way of performing software development and data analyses as it enables users to take advantage of a completely automated provenance tracking which happens silently in the background.

Application Definitions

Application definitions for the input side (configuration) of each tool were defined.

***NXapm_paraprobe_config_transcoder*:**
Configuration of paraprobe-transcoder Load POS, ePOS, APSuite APT, RRNG, RNG, and NXapm HDF5 files.

***NXapm_paraprobe_config_ranger*:**
Configuration of paraprobe-ranger Apply ranging definitions and explore possible molecular ions.

***NXapm_paraprobe_config_selector*:**
Configuration of paraprobe-selector Defining complex spatial regions-of-interest to filter reconstructed datasets.

***NXapm_paraprobe_config_surfacer*:**
Configuration of paraprobe-surfacer Create a model for the edge of a point cloud via convex hulls, alpha shapes.

***NXapm_paraprobe_config_distancer*:**
Configuration of paraprobe-distancer Compute analytical distances between ions to a set of triangles.

***NXapm_paraprobe_config_tessellator*:**
Configuration of paraprobe-tessellator Compute Voronoi cells for if desired all ions in a dataset.

***NXapm_paraprobe_config_nanochem*:**
Configuration of paraprobe-nanochem Compute delocalization, iso-surfaces, analyze 3D objects, and composition profiles.

***NXapm_paraprobe_config_intersector*:**
Configuration of paraprobe-intersector Assess intersections and proximity of 3D triangulated surface meshes in continuum space to study the effect the parameterization of surface extraction algorithms on the resulting shape, spatial arrangement, and colocation of 3D objects via graph-based techniques.

***NXapm_paraprobe_config_spatstat*:**
Configuration of paraprobe-spatstat Spatial statistics on the entire or selected regions of the reconstructed dataset.

***NXapm_paraprobe_config_clusterer*:**
Configuration of paraprobe-clusterer Import cluster analysis results of IVAS/APSuite and perform clustering analyses in a Python ecosystem.

Application definitions for the output side (results) of each tool were defined.

***NXapm_paraprobe_results_transcoder*:**
Results of paraprobe-transcoder Store reconstructed positions, ions, and charge states.

***NXapm_paraprobe_results_ranger*:**
Results of paraprobe-ranger Store applied ranging definitions and combinatorial analyses of all possible iontypes.

NXapm_paraprobe_results_selector:
Results of paraprobe-selector Store which points are inside or on the boundary of complex spatial regions-of-interest.

NXapm_paraprobe_results_surfacer:
Results of paraprobe-surfacer Store triangulated surface meshes of models for the edge of a dataset.

NXapm_paraprobe_results_distancer:
Results of paraprobe-distancer Store analytical distances between ions to a set of triangles.

NXapm_paraprobe_results_tessellator:
Results of paraprobe-tessellator Store volume of all Voronoi cells about each ion in the dataset.

NXapm_paraprobe_results_nanochem:
Results of paraprobe-nanochem Store all results of delocalization, isosurface, and interface detection algorithms, store all extracted triangulated surface meshes of found microstructural features, store composition profiles and corresponding geometric primitives (ROIs).

NXapm_paraprobe_results_intersector:
Results of paraprobe-intersector Store graph of microstructural features and relations/link identified between them.

NXapm_paraprobe_results_spatstat:
Results of paraprobe-spatstat Store spatial correlation functions.

NXapm_paraprobe_results_clusterer:
Results of paraprobe-clusterer Store results of cluster analyses.

Base Classes

We envision that the above-mentioned definitions can be useful not only to take inspiration for other software tools in the field of atom probe but also to support a discussion towards a stronger standardization of the vocabulary used. Therefore, we are happy for comments and suggestions.

The majority of data analyses in atom probe use a common set of operations and conditions on the input data even though this might not be immediately evident or might not have been so explicitly communicated in the literature. Some tools have a specific scope because of which algorithms are hardcoded to work for specific material systems. A typical example is a ranging tool which exploits that all the examples it is used for apply to a specific material and thus specific iontypes can be hardcoded.

Instead, we are convinced it is better to follow a more generalized approach. The following base classes and the above application definitions present examples how one can use NeXus for this.

NXapm_input_reconstruction:
A description from which file the reconstructed ion positions are imported.

NXapm_input_ranging:
A description from which file the ranging definitions are imported. The design of the ranging definitions is, thanks to *NXion*, so general that all possible nuclids can be considered, be they observationally stable, be they radioactive or transuranium nuclids.

A detailed inspection of spatial and other type of filters frequently used in analysis of atom probe data revealed that it is better to define atom-probe-agnostic, i.e. more general filters:

NXspatial_filter:
A proposal how a point cloud can be spatially filtered in a specific yet general manner. This base class takes advantage of *NXcg_ellipsoid_set*, *NXcg_cylinder_set*, and *NXcg_hexahedron_set* to cater for all of the most commonly used geometric primitives in atom probe.

NXsubsampling_filter:
A proposal for a filter that can also be used for specifying how entries like ions can be filtered via sub-sampling.

NXmatch_filter:
A proposal for a filter that can also be used for specifying how entries like ions can be filtered based on their type (ion species) or hit multiplicity.

Transport Phenomena

Introduction

Application Definitions

We realize that many experiments in condensed-matter physics and materials engineering belong to the category of measurements of transparent phenomena. A possible example of such experiments are temperature-dependent current-voltage (IV) curve measurements (or JV for engineers) measurements. In this case, electrical charge is transported and the temperature-dependent current response as a function of applied voltage is recorded.

Below is an example for such an application definition for an experiment. This application definition has exemplar parts which show how such an experiment can be controlled with the EPICS system:

NXiv_temp:
Application definition for temperature-dependent current-voltage (IV) curve measurements.

Geometry & Microstructure

Introduction

The computational-geometry/microstructure-modeling-based part of the proposal has the following aims:

First, we would like to contribute to efforts on standardizing a controlled vocabulary, definitions for these terms, and relations between the terms, for computational-geometry-based descriptions of the structure of materials and atomic configurations used when characterizing materials in experiments and computer simulations.

As far as NeXus is concerned, this proposed set of simple geometric primitives and shapes offer a complementary alternative to the current set of base classes in NeXus for constructive solid geometry such as *NXcsg*, *NXoff_geometry*, or *NXquadric* to name but a few.

Second, we would like to explore with this proposal how we can harmonize terms frequently used by materials scientists in the field of condensed-matter physics with definitions and terms offer by the NOMAD MetaInfo description.

Third, the proposal should yield a substantiated set of arguments and suggestions how descriptors for the structure and atomic architecture of materials can be harmonized. With this we especially would like to reach out and intensify the cooperation between the materials-science-related consortia of the German National Research Infrastructure, specifically, FAIRmat, NFDI-MatWerk, NFDI4Ing, NFDI4Chem, NFDI4Cat, MaRDi, and DAPHNE.

Physics background

Microstructural features or crystal defects are spatial arrangements of atoms. Given their specific atomic arrangement and composition, such features have specific constraints on the degrees of freedom how atoms can arrange. This causes these defects to have specific properties. Provided well-defined coarse-graining procedures are used and regions-of-interest and/or regions-of-applicability are defined, microstructural features are often characterized and modelled to have associated thermodynamic descriptors.

Another motivation for the proposal was the observation that frequently the design of file formats for simulation software in the computational materials science especially those tools at the interface between condensed-matter physics and materials engineering are frequently reimplementing the wheel (at least partly) when it comes to decide how to store e.g. atom and feature positions or shape of regions-of-interest, grids, crystals, grains, precipitates, and dislocations.

Maybe this is a historical burden given the large set of technical terms and jargon in place, which then motivated pragmatic solutions that resulted in many research groups have developed similar formats using similar descriptions.

We see this work on base classes and application definitions not primarily an effort to improve and extend NeXus for now. Rather this part of the proposal is an effort to support activities in materials science to work towards common terminology and using controlled vocabularies more frequently. These are the foundation for more sophisticated thoughts about practically useful ontologies.

Defining crystal defects is a question of how to coarse-grain a given spatiotemporal set of atoms, each having a nuclide type and position/trajectory. Different mathematical/geometrical methods exists to coarse-grain and thus determine how a point, a line, a surface, or a volumetric defect can be described and be spatiotemporally constrained through a geometrical model with defined geometric primitives and associated (materials) properties at a coarser-scale.

The key motivation to such coarse-graining is to reduce the complexity of the description. On the one hand to support visualization and scientific analyses - not only of crystal defect arrangements. On the other hand it is the hope that using descriptors at a coarser level, i.e. nanometre, micrometre, and larger, it is still possible to find sufficiently accurate and precise descriptors that avoid one having to account for the dynamics of each atom to predict or understand the properties of defects and their dynamics.

Nevertheless, experience has shown that computational-geometry-based descriptions when combined with hierarchical clustering/labeling methods, applied on set of atoms and features turn out to yield useful descriptors. Examples include point, line, surface defects, such as vacancies, solute cluster, dislocations, disconnections, interfaces to name but a few.

Base Classes

The following base classes are defined to incentivize the use of NeXus for using computational-geometry-based descriptions. In what follows, base classes for frequently used shapes and geometric primitives are proposed:

NXcg_sphere_set:
    A description for a set of possibly dissimilar spheres.

NXcg_ellipsoid_set:
    A description for a set of possibly dissimilar oriented ellipsoids.

NXcg_cylinder_set:
    A description for a set of possibly dissimilar oriented cylinders.

NXcg_point_set:
    A collection of points with labels.

NXcg_polyline_set:
    A collection of lines and linear segments.

NXcg_triangle_set:
    A collection of triangles.

NXcg_parallelogram_set:
A collection of possibly dissimilar parallelograms.

NXcg_triangulated_surface_mesh:
A mesh of triangles.

NXcg_polygon_set:
A collection of polygons.

NXcg_polyhedron_set:
A collection of polyhedra.

NXcg_roi_set:
A container to host a number of different types of primitives.

NXcg_tetrahedron_set:
A collection of tetrahedra.

NXcg_hexahedron_set:
A collection of hexahedra with capabilities to represent also simpler (bounding) boxes for e.g. binary trees.

These base classes describe data structures used for more complex geometries:

NXcg_face_list_data_structure:
In essence, the usual way how polygon/polyhedra data are reported: A list of vertices and faces with identifier and properties.

NXcg_half_edge_data_structure:
A half-edge data structure (also known as a doubly connected edge list) is a useful complementary descriptor for polygon/polyhedra which enables topological analyses and traversal of the graph of how polygons and polyhedra are connected.

NXcg_unit_normal_set:
As an additional structuring element especially for meshes, well-documented normal information is crucial for distance computations.

NXcg_geodesic_mesh:
Geodesic meshes are useful for all applications when meshing the surface of a sphere.

NXcg_alpha_complex:
Alpha shapes and alpha wrappings, specifically the special case of the convex hull, are frequently used geometrical models for describing a boundary or edge to a set of geometric primitives.

Next, a few base classes are defined for documenting discretized representations of material (area or volume) which can be useful not only for stencil-based methods:

NXcg_grid:
A grid of cells.

NXisocontour:
A description for isocontour descriptions.

NXcg_marching_cubes:
An approach to store metadata of a specific implementation of the Marching Cubes algorithm, whose sensitivity to specific topological configurations is known to result in different triangle collections.

NXdelocalization:
An approach to document procedures whereby a scalar field is smoothed in a controlled manner.

NXsimilarity_grouping:
An alternative for NXclustering.

NXclustering:
A description for clustering of objects (such as atoms or features).

NXorientation_set:
A set of parameters to describe the relative orientation of members of a set of features/objects.

NXslip_system_set:
Metadata for a set of slip systems in a given crystal structure.

NXms_feature_set:
Generic base class to describe any nested set of features of a microstructure at the continuum-, micron-, nano-scale, or to represent a topology of molecules and atoms.

NXms_snapshot:
A container to describe the state of microstructural features at a given point in time.

NXms_snapshot_set:
The corresponding class to hold a set of *NXms_snapshot* objects.

NXchemical_composition:
(Chemical) composition of a sample or a set of things.

Finally, the following base classes allow data processing software to document its input parameters and to summarize its performance statistics:

NXprogram:
A named and version of a program of library/component.

NXcs_filter_boolean_mask:
A boolean mask.

NXcs_prng:
Metadata of a pseudo-random number generator (PRNG) algorithm.

NXcs_profiling:
A structuring group holding a set of *NXcs_profiling_event* instances.

NXcs_profiling_event:
Profiling/benchmark data to an event of tracking an algorithm/computational step.

NXcs_computer:
Metadata of a computer.

NXcs_cpu:
Metadata of a central processing unit.

NXcs_gpu:
Metadata of a graphical processing unit / accelerator.

NXcs_mm_sys:
Metadata of the (main) memory (sub-)system.

NXcs_io_sys:
Metadata of the input/output system.

NXcs_io_obj:
Metadata of a component storing data of an *NXcs_io_sys* instance.

Application definitions for ICME models

It is important to embrace the large research community of materials engineers as they are frequent users of electron microscopy and atom probe microscopy. In this community frequently ICME (Integrated Computational Materials Engineering) microstructure models are used. These models are derived from a design strategy and workflow whereby physics-based modelling of microstructure evolution, typically at the mesoscopic scale, is used to understand the relations between the microstructure and technological relevant descriptors for the properties of materials.

The following application definitions are proposed to support discussion on how materials engineering-specific data models connect to or can be mapped on concepts which are equally modellable with NeXus:

***NXms*:**
An application definition for arbitrary spatiotemporally resolved simulations.

***NXms_score_config*:**
A specific example of how *NXapm_paraprobe_config_ranger* can be specialized for documenting the configuration of a computer simulation with the static recrystallization cellular automata model SCORE.

***NXms_score_results*:**
A specific example of how *NXms* can be specialized for documenting results of computer simulations with the static recrystallization cellular automata model SCORE.

NXaberration

Status:

base class, extends *NXobject*

Description:

Quantified aberration coefficient in an aberration_model.

Symbols:

No symbol table

Groups cited:
none

Structure:

magnitude: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

uncertainty: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Confidence

uncertainty_model: (optional) *NX_CHAR*

How was the uncertainty quantified e.g. via the 95% confidence interval.

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*}

Time elapsed since the last measurement.

angle: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

For the CEOS definitions the C aberrations are radial-symmetric and have no angle entry, while the A, B, D, S, or R aberrations are n-fold symmetric and have an angle entry. For the NION definitions the coordinate system differs to the one used in CEOS and instead two aberration coefficients a and b are used.

name: (optional) *NX_CHAR*

alias: (optional) *NX_CHAR*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXaberration/alias-field*
- */NXaberration/angle-field*
- */NXaberration/delta_time-field*
- */NXaberration/magnitude-field*
- */NXaberration/name-field*
- */NXaberration/uncertainty-field*
- */NXaberration/uncertainty_model-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXaberration.nxdl.xml

NXaberration_model

Status:

base class, extends *NXobject*

Description:

Models for aberrations of electro-magnetic lenses in electron microscopy.

See S. J. Pennycock and P. D. Nellist (page 44ff, and page 118ff) for different definitions available and further details. Table 7-2 of Ibid. publication (page 305ff) documents how to convert from the NION to the CEOS definitions.

Symbols:

No symbol table

Groups cited:
NXaberration

Structure:

model: (optional) *NX_CHAR*

Any of these values: `ceos` | `nion`

c_1_0: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Defocus

c_1_2_a: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Two-fold astigmatism

c_1_2_b: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Two-fold astigmatism

c_2_1_a: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Second-order axial coma

c_2_1_b: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Second-order axial coma

c_2_3_a: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Threefold astigmatism

c_2_3_b: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Threefold astigmatism

c_3_0: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Spherical aberration

c_3_2_a: (optional) *NX_FLOAT* {units=*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

ABERRATION: (optional) *NXaberration*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXaberration_model/ABERRATION-group*
- */NXaberration_model/c_1_0-field*
- */NXaberration_model/c_1_2_a-field*
- */NXaberration_model/c_1_2_b-field*
- */NXaberration_model/c_2_1_a-field*
- */NXaberration_model/c_2_1_b-field*
- */NXaberration_model/c_2_3_a-field*
- */NXaberration_model/c_2_3_b-field*
- */NXaberration_model/c_3_0-field*
- */NXaberration_model/c_3_2_a-field*
- */NXaberration_model/c_3_2_b-field*
- */NXaberration_model/c_3_4_a-field*
- */NXaberration_model/c_3_4_b-field*

- */NXaberration_model/c_5_0-field*
- */NXaberration_model/model-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXaberration_model.nxdl.xml

NXaberration_model_ceos

Status:

base class, extends *NXobject*

Description:

CEOS definitions/model for aberrations of electro-magnetic lenses.

See S. J. Pennycock and P. D. Nellist (page 44ff, and page 118ff) for different definitions available and further details. Table 7-2 of Ibid. publication (page 305ff) documents how to convert from the NION to the CEOS definitions.

Symbols:

No symbol table

Groups cited:
NXaberration

Structure:

model: (optional) *NX_CHAR*

Obligatory value: `ceos`

c_1: (optional) *NXaberration*

a_1: (optional) *NXaberration*

b_2: (optional) *NXaberration*

a_2: (optional) *NXaberration*

c_3: (optional) *NXaberration*

s_3: (optional) *NXaberration*

a_3: (optional) *NXaberration*

b_4: (optional) *NXaberration*

d_4: (optional) *NXaberration*

a_4: (optional) *NXaberration*

c_5: (optional) *NXaberration*

s_5: (optional) *NXaberration*

r_5: (optional) *NXaberration*

a_6: (optional) *NXaberration*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXaberration_model_ceos/a_1-group*
- */NXaberration_model_ceos/a_2-group*
- */NXaberration_model_ceos/a_3-group*
- */NXaberration_model_ceos/a_4-group*
- */NXaberration_model_ceos/a_6-group*
- */NXaberration_model_ceos/b_2-group*
- */NXaberration_model_ceos/b_4-group*
- */NXaberration_model_ceos/c_1-group*
- */NXaberration_model_ceos/c_3-group*
- */NXaberration_model_ceos/c_5-group*
- */NXaberration_model_ceos/d_4-group*
- */NXaberration_model_ceos/model-field*
- */NXaberration_model_ceos/r_5-group*
- */NXaberration_model_ceos/s_3-group*
- */NXaberration_model_ceos/s_5-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXaberration_model_ceos.nxdl.xml

NXaberration_model_nion

Status:

base class, extends *NXobject*

Description:

NIONdefinitions/model for aberrations of electro-magnetic lenses.

See S. J. Pennycock and P. D. Nellist (page 44ff, and page 118ff) for different definitions available and further details. Table 7-2 of Ibid. publication (page 305ff) documents how to convert from the NION to the CEOS definitions.

Symbols:

No symbol table

Groups cited:
NXaberration

Structure:

model: (optional) *NX_CHAR*

Obligatory value: `nion`

c_1_0: (optional) *NXaberration*

c_1_2_a: (optional) *NXaberration*

c_1_2_b: (optional) *NXaberration*

c_2_1_a: (optional) *NXaberration*

c_2_1_b: (optional) *NXaberration*

c_2_3_a: (optional) *NXaberration*

c_2_3_b: (optional) *NXaberration*

c_3_0: (optional) *NXaberration*

c_3_2_a: (optional) *NXaberration*

c_3_2_b: (optional) *NXaberration*

c_3_4_a: (optional) *NXaberration*

c_3_4_b: (optional) *NXaberration*

c_4_1_a: (optional) *NXaberration*

c_4_1_b: (optional) *NXaberration*

c_4_3_a: (optional) *NXaberration*

c_4_3_b: (optional) *NXaberration*

c_4_5_a: (optional) *NXaberration*

c_4_5_b: (optional) *NXaberration*

c_5_0: (optional) *NXaberration*

c_5_2_a: (optional) *NXaberration*

c_5_2_b: (optional) *NXaberration*

c_5_4_a: (optional) *NXaberration*

c_5_4_b: (optional) *NXaberration*

c_5_6_a: (optional) *NXaberration*

c_5_6_b: (optional) *NXaberration*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXaberration_model_nion/c_1_0-group*
- */NXaberration_model_nion/c_1_2_a-group*
- */NXaberration_model_nion/c_1_2_b-group*
- */NXaberration_model_nion/c_2_1_a-group*
- */NXaberration_model_nion/c_2_1_b-group*
- */NXaberration_model_nion/c_2_3_a-group*
- */NXaberration_model_nion/c_2_3_b-group*
- */NXaberration_model_nion/c_3_0-group*

- */NXaberration_model_nion/c_3_2_a-group*
- */NXaberration_model_nion/c_3_2_b-group*
- */NXaberration_model_nion/c_3_4_a-group*
- */NXaberration_model_nion/c_3_4_b-group*
- */NXaberration_model_nion/c_4_1_a-group*
- */NXaberration_model_nion/c_4_1_b-group*
- */NXaberration_model_nion/c_4_3_a-group*
- */NXaberration_model_nion/c_4_3_b-group*
- */NXaberration_model_nion/c_4_5_a-group*
- */NXaberration_model_nion/c_4_5_b-group*
- */NXaberration_model_nion/c_5_0-group*
- */NXaberration_model_nion/c_5_2_a-group*
- */NXaberration_model_nion/c_5_2_b-group*
- */NXaberration_model_nion/c_5_4_a-group*
- */NXaberration_model_nion/c_5_4_b-group*
- */NXaberration_model_nion/c_5_6_a-group*
- */NXaberration_model_nion/c_5_6_b-group*
- */NXaberration_model_nion/model-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXaberration_model_nion.nxdl.xml

NXadc

Status:

base class, extends *NXobject*

Description:

Analog-to-digital converter component/integrated circuit.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
none

Structure:

value: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

TBD.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXadc/value-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXadc.nxdl.xml

NXaperture_em

Status:

base class, extends *NXobject*

Description:

Details of an individual aperture for beams in electron microscopy.

Symbols:

No symbol table

Groups cited:
NXfabrication, *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.

FABRICATION: (optional) *NXfabrication*

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/FABRICATION-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 and field ion microscopy experiments.

This application definition provides a place to document data and metadata to an atom probe experiment. Primarily the measurement itself is documented. However, as most atom probe experiments are controlled with commercial software which does not allow to access the raw detector hits, this application definition also includes two key groups of processing steps (reconstruction and ranging).

During tomographic reconstruction measured data are processed into a point cloud of reconstructed positions of certain ions. During ranging time-of-flight data are identified as representing specific ions to annotate each ion with a label.

Commercial software used in atom probe research is designed as an integrated acquisition and instrument control software. For AMETEK/Cameca local electrode atom probe (LEAP) instruments the least processed (rawest) numerical results and metadata are stored in so-called STR, RRAW, RHIT, and HITS files, which are proprietary and their file format specifications not publicly documented.

Supplementary metadata are kept in a database (formerly known as the ISDb) which is connected to the instrument control software and synced with the experiment while ions are detected. In effect, RHIT and HITS files store the (rawest) experiment data 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 (APM) 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) and 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. This is post-processing even though in practice it is performed sometimes already (as preview) already while data are still being collected.

The ion types decode molecular identities which can very often be mapped to elemental identities, and also be used to distinguish 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 stage of data acquisition and processing stages to obtain a useful reconstructed and ranged dataset.

This 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.

Future guidance by the technology partners like AMETEK/Cameca could improve this description to cover a substantial larger number of eventually metadata that so far are neither publicly documented nor accessible.

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 intervals of 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.

n_topology: Total number of integers in the supplementary XDMF topology array.

Groups cited:
NXaperture_em, *NXbeam*, *NXchamber*, *NXchemical_composition*, *NXcollection*, *NXcoordinate_system_set*, *NXcsg*, *NXdata*, *NXdetector*, *NXentry*, *NXfabrication*, *NXinstrument*, *NXion*, *NXlens_em*, *NXmonitor*, *NXnote*, *NXpeak*, *NXprocess*, *NXprogram*, *NXpulser_apm*, *NXpump*, *NXreflectron*, *NXsample*, *NXsource*, *NXstage_lab*, *NXtransformations*, *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 the application definition behind instead of filling in these details into the experiment_description free-text description field.

Users are encouraged to add in this field eventual DOIs to papers which yield further details to the experiment.

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 dates specified, it may not be possible to infer how long the experiment took or for how long data were collected.

More detailed timing data over the course of the experiment have to be collected to compute this event chain during the experiment.

end_time: (recommended) *NX_DATE_TIME <=*

ISO 8601 time code with local time zone offset to UTC included when the microscope session ended.

run_number: (recommended) *NX_CHAR*

Neither the specimen_name nor 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/APSuite run_number. For other instruments, such as the one from Stuttgart or Oxcart from Erlangen, or the instruments at GPM in Rouen, use the identifier which is closest in meaning to the LEAP run number. The field does not have to be required if the information is recoverable in the dataset which for LEAP instruments is the case when RHIT or HITS files are also stored alongside a data artifact instance which is generated according to this NXapm application definition.

As a destructive microscopy technique, 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.

This application definition does currently not allow storing the entire set of such interrupted runs. Not because of a technical limitation within NeXus but because we have not seen a covering use case based on which we could have designed and implemented this case. Atom probers are invited to contact the respective people in the FAIRmat team to fix this.

operation_mode: (required) *NX_CHAR*

What type of atom probe microscopy experiment is performed. This field is primarily meant to inform materials database systems to qualitatively filter experiments:

- apt are experiments where the analysis_chamber has no imaging gas. experiment with LEAP instruments are typically performed as apt.

- fim are experiments where the analysis_chamber has an imaging gas, which should be specified with the atmosphere in the analysis_chamber group.
- apt_fim should be used for combinations of the two imaging modes.
- other should be used in combination with the user specifying details in the experiment_documentation field.

Any of these values: `apt` | `fim` | `apt_fim` | `other`

PROGRAM: (required) *NXprogram*

program: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

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 pieces of information should be mapped to respective groups and sections.

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 <=*

USER: (optional) *NXuser <=*

Contact information and eventually details person(s) involved in 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: (recommended) *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. If this field is field the specific service should also be written in orcid_platform

orcid_platform: (recommended) *NX_CHAR <=*

Name of the OrcID or ResearcherID where the account under orcid is registered.

telephone_number: (optional) *NX_CHAR* *<=*

(Business) (tele)phone number of the user at the point in time when the experiment was performed.

role: (recommended) *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: (recommended) *NXsample* *<=*

Description of the sample from which the specimen was prepared or site-specifically cut out using e.g. a focused-ion beam instrument.

The sample group is currently a place for storing suggestions from atom probers about other knowledge they have gained about the sample from which they cut the specimen which is field-evaporated during the experiment. Typically this is possible because the atom probe specimen is usually not heat treated as is but one assumes that one has the sample prepared as needed (i.e. with a specific grain diameter) and can thus just cut out the specimen from that material.

There are cases though where the specimen is processed further, i.e. the specimen is machined further or exposed to external stimuli during the experiment. In this case, these details should not be stored in the sample group but atom probers should make suggestions how this application definition can be improved to find a better place and compromise how to improve this application definition.

In the future also details like how the grain_diameter was characterized, how the sample was prepared, how the material was heat-treated etc., should be stored as using specific application definitions/schemas which are then arranged and documented with a description of the workflow so that actionable graphs become instantiatable.

grain_diameter: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Qualitative information about the grain size, here specifically described as the equivalent spherical diameter of an assumed average grain size for the crystal ensemble. Users of this information should be aware that although the grain diameter or radius is often referred to as grain size and used in e.g. Hall-Petch-type materials models this is if at all an ensemble average whose reporting may be very informative or not if the specimen contains a few grains only. In atom probe the latter is often the case because grains are measured partially as their diameter can be in the order of magnitude of the physical diameter of the specimen.

Reporting a grain size is useful though as it allows judging if specific features are expected to be found in the detector hit map.

grain_diameter_error: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Magnitude of the standard deviation of the grain_diameter.

heat_treatment_temperature: (optional) *NX_FLOAT* {units=*NX_TEMPERATURE*}

The temperature of the last heat treatment step before quenching. Knowledge about this value can give an idea how the sample was heat treated, however if available a documentation of the annealing treatment should be delivered by adding additional files which are uploaded alongside an NXapm instance. In the future there should better be an own schema used for the heat treatment.

heat_treatment_temperature_error: (optional) *NX_FLOAT* {units=*NX_TEMPERATURE*}

Magnitude of the standard deviation of the heat_treatment_temperature.

heat_treatment_quenching_rate: (optional) *NX_FLOAT* {units=*NX_ANY*}

Rate of the last quenching step. Knowledge about this value can give an idea how the specimen was heat treated, however there are many situations where one can imagine that the scalar value for just the quenching rate, i.e. the first derivative of the measured time-temperature profile is itself time-dependant. An example is when the specimen was left in the furnace after the furnace was switched off. In this case the specimen cools down with a specific rate of how this furnace cools down in the lab. Processes which in practice are often not documented with measuring the time-temperature profile.

This can be problematic because when the furnace door was left open or the ambient temperature in the lab changes, i.e. for a series of experiments where one is conducted on a hot summer day and the next during winter as might have an effect on the evolution of the microstructure. There are many cases where this has been reported to be an issue in industry, e.g. think about aging aluminium samples left on the factory parking lot on a hot summer day.

heat_treatment_quenching_rate_error: (optional) *NX_FLOAT* {units=*NX_ANY*}

Magnitude of the standard deviation of the heat_treatment_quenching_rate.

description: (optional) *NX_CHAR* <=

CHEMICAL_COMPOSITION: (recommended) *NXchemical_composition*

The chemical composition of the sample. Typically it is assumed that this more macroscopic composition is representative for the material so that the composition of the typically substantially less voluminous specimen probes from the more voluminous sample. **normalization**: (required) *NX_CHAR*

Reporting compositions as atom and weight percent yields both dimensionless quantities but their conceptual interpretation differs. A normalization based on atom_percent counts relative to the total number of atoms are of a particular type. By contrast, weight_percent normalization factorizes in the respective mass of the elements. Python libraries like pint are challenged by these differences as at.-% and wt.-% both yield fractional quantities.

Any of these values: `atom_percent` | `weight_percent`

ION: (required) *NXion* <=

name: (required) *NX_CHAR* <=

Human-readable name of the element/ion (e.g. Fe). Name has to be a symbol of an element from the periodic table. All symbols in the set of NXion instances inside the group chemical_composition need to be disjoint.

composition: (required) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Composition value for the element/ion referred to under name. The value is normalized based on normalization, i.e. composition is either an atom or weight percent quantity.

composition_error: (optional) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Magnitude of the standard deviation of the composition (value).

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 (see the sample group) from where this specimen was cut. In cases where the specimen was e.g. site-specifically cut from the sample referred to in the sample group 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 a specimen was cut/prepared from a specific sample in the sample_history. The sample_history field must not be used for writing an alias of the specimen. Instead, use the field alias for this. As an example there may be a specimen/sample monitoring system in a lab with bar codes. The bar code is a good specimen/sample name. A shorter and more human readable alias like A0 can be an example for something to write in the alias field.

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 is to be used for the characterization of a single specimen in a single continuous run.

Details about the specimen preparation should be stored in the sample_history or if this is not possible in the sample group.

sample_history: (recommended) *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 as 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: (recommended) *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.

alias: (optional) *NX_CHAR*

Short_name or abbreviation of the specimen name field.

atom_types: (required) *NX_CHAR*

List of comma-separated elements from the periodic table that are contained in the sample. If the sample substance has multiple components, all elements from each component must be included in *atom_types*.

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 or from other data sources.

description: (optional) *NX_CHAR* <=

Discouraged free-text field in case properly designed records for the sample_history or sample section are not available.

is_polycrystalline: (recommended) *NX_BOOLEAN*

Report if the specimen is polycrystalline, in which case it contains a grain or phase boundary, or if the specimen is a single crystal.

DATA: (optional) *NXdata* <=

Hard link to a location in the hierarchy of the NeXus file where the data for default plotting are stored.

COORDINATE_SYSTEM_SET: (recommended) *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):

- World space; a CS specifying a local coordinate system of the planet earth which identifies into which direction gravity is pointing such that the laboratory space CS can be rotated into this world 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 has to be pushed during loading a specimen into the instrument. In summary, this CS is defined by the chassis of 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 to the detector plane whose xy plane is usually in the detector and whose z axis points towards the specimen. This is a distorted space with respect to the reconstructed ion positions.
- The reconstruction space; a CS in which the reconstructed ion positions are defined. The orientation depends on the analysis 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. **TRANSFORMATIONS**: (optional) *NXtransformations* *<=*

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 the 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 along which ions pass nominally when they leave the specimen and travel to the detector.

THIS DOCSTRING NEEDS CLARIFICATION.

field_of_view: (recommended) *NX_FLOAT* {units=*NX_LENGTH*}

The nominal diameter of the specimen ROI which is measured in the experiment. It is important to mention that the physical specimen cannot be measured completely because ions may launch but not be detected or hit elsewhere in the analysis_chamber.

status: (recommended) *NX_CHAR*

A statement whether the measurement was successful or failed prematurely.

Any of these values: `success` | `failure` | `unknown`

CSG: (optional) *NXcsg*

FABRICATION: (recommended) *NXfabrication*

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

REFLECTRON: (required) *NXreflectron*

applied: (required) *NX_BOOLEAN*

Is a reflectron installed and was it used?

name: (optional) *NX_CHAR* <=

description: (recommended) *NX_CHAR* <=

FABRICATION: (optional) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CSG: (optional) *NXcsg*

local_electrode: (required) *NXlens_em*

A local electrode guiding the ion flight path. Also called counter or extraction electrode. **name**: (required) *NX_CHAR* <=

Identifier of the local_electrode in an e.g. database.

APERTURE_EM: (optional) *NXaperture_em*

name: (recommended) *NX_CHAR* <=

value: (recommended) *NX_NUMBER* <=

FABRICATION: (optional) *NXfabrication* <=

identifier: (recommended) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CSG: (optional) *NXcsg*

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.

CSG: (optional) *NXcsg*

pulser: (required) *NXpulser_apm*

pulse_mode: (required) *NX_CHAR* *<=*

pulse_frequency: (required) *NX_FLOAT* *<=*

pulse_fraction: (required) *NX_FLOAT* *<=*

pulsed_voltage: (recommended) *NX_FLOAT* *<=*

standing_voltage: (recommended) *NX_FLOAT* *<=*

CSG: (optional) *NXcsg*

SOURCE: (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 experiment. If pulse_mode is set to laser or laser_and_voltage (e.g. for LEAP6000-type instruments) having the group/section laser_gun is required and the following of its fields have to be filled:

- name
- wavelength
- energy

name: (required) *NX_CHAR* *<=*

wavelength: (recommended) *NX_FLOAT* *<=*

power: (recommended) *NX_FLOAT* <=

pulse_energy: (recommended) *NX_FLOAT* <=

FABRICATION: (optional) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

BEAM: (optional) *NXbeam* <=

pinhole_position: (recommended) *NXcollection* <=

spot_position: (recommended) *NXcollection* <=

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.

temperature: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_TEMPERATURE*}

The best estimate, at experiment time, for the temperature at the sample base (furthest point along sample apex and holding assembly that is removable from the sample stage).

CSG: (optional) *NXcsg*

analysis_chamber: (required) *NXchamber*

name: (optional) *NX_CHAR* <=

description: (optional) *NX_CHAR* <=

pressure: (required) *NX_FLOAT* {units=*NX_PRESSURE*}

Average pressure in the analysis chamber.

FABRICATION: (optional) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CSG: (optional) *NXcsg*

buffer_chamber: (optional) *NXchamber*

name: (optional) *NX_CHAR* <=

description: (optional) *NX_CHAR* <=

pressure: (required) *NX_FLOAT* {units=*NX_PRESSURE*}

Average pressure in the buffer chamber.

FABRICATION: (optional) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CSG: (optional) *NXcsg*

load_lock_chamber: (optional) *NXchamber*

name: (optional) *NX_CHAR* <=

description: (optional) *NX_CHAR* <=

pressure: (required) *NX_FLOAT* {units=*NX_PRESSURE*}

Average pressure in the load_lock_chamber.

FABRICATION: (optional) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CSG: (optional) *NXcsg*

getter_pump: (optional) *NXpump*

design: (recommended) *NX_CHAR* <=

FABRICATION: (optional) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CSG: (optional) *NXcsg*

roughening_pump: (optional) *NXpump*

design: (recommended) *NX_CHAR* <=

FABRICATION: (optional) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CSG: (optional) *NXcsg*

turbomolecular_pump: (optional) *NXpump*

design: (recommended) *NX_CHAR* <=

FABRICATION: (optional) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* *<=*

model: (recommended) *NX_CHAR* *<=*

identifier: (recommended) *NX_CHAR* *<=*

capabilities: (optional) *NX_CHAR*

CSG: (optional) *NXcsg*

instrument_calibration: (recommended) *NXcollection* *<=*

A possible place, which has to be discussed with the atom probe community more though, where quantitative details about the calibration of the counter electrode could be stored. Work in this direction was e.g. reported by the Erlangen group (see P. Felfer et al.)

specimen_monitoring: (recommended) *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 method, which is less accurate 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. Nothing can be said about the outcome of this research yet but here is where such spatio-temporally data could be stored.

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.

detection_rate: (required) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Average detection rate over the course of the experiment.

estimated_field_at_the_apex: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_ANY*}

Estimated field at the apex along the evaporation sequence.

control_software: (required) *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) (APSuite/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 Felfer (Oxcart, Erlangen, Germany), the Northwestern (D. Isheim, Seidman group et al.), or the PNNL group (Pacific Northwest National Laborary, 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. **PROGRAM**: (required) *NXprogram*

program: (required) *NX_CHAR <=*

@**version**: (required) *NX_CHAR <=*

buffer_chamber: (optional) *NXcollection*

Track time-dependent details over the course of the measurement about the buffer_chamber.

load_lock_chamber: (optional) *NXcollection*

Track time-dependent details over the course of the measurement about the load_lock_chamber.

analysis_chamber: (optional) *NXcollection*

Track time-dependent details over the course of the measurement about the analysis_chamber.

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. **sequence_index**: (recommended) *NX_POSINT <=*

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). If the acquisition software enables it one can also store in this field the hit_positions, as measured by the detector, without any corrections.

PROGRAM: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

hit_quality_filtering: (optional) *NXprocess*

This could be a place where currently the publicly undocumented algorithmic steps are stored how detected hits are judged for their quality. In CamecaRoot this there is something mentioned like golden and partial hits, here is where this could be documented. **sequence_index**: (recommended) *NX_POSINT* <=

hit_multiplicity: (recommended) *NXprocess*

Data post-processing step which is, like the impact position analyses, 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 f the same element or isotope, respectively, a molecular ion contains (which is instead encoded with the isotope_vector field of each NXion instance). **sequence_index**: (recommended) *NX_POSINT* <=

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.

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.

hit_multiplicity: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*}

Hit multiplicity.

PROGRAM: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

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. **sequence_index**: (recommended) *NX_POSINT* <=

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.

PROGRAM: (required) *NXprogram*

program: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

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. **sequence_index**: (recommended) *NX_POSINT <=*

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.

PROGRAM: (required) *NXprogram*

program: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

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 instruments 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 see displayed in e.g. APSuite 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. **sequence_index**: (recommended) *NX_POSINT <=*

mass_to_charge: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_ANY*}

Mass-to-charge-state ratio values.

PROGRAM: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

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.
sequence_index: (recommended) *NX_POSINT* <=

protocol_name: (required) *NX_CHAR*

Qualitative statement about which reconstruction protocol was used.

Any of these values: `bas` | `geiser` | `gault` | `cameca` | `other`

parameter: (required) *NX_CHAR*

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.

crystallographic_calibration: (required) *NX_CHAR*

Different strategies for crystallographic calibration of the reconstruction are possible. The field is required and details should be specified in free-text at least. If the not crystallographic calibration was performed the field should be filled with the n/a, meaning not applied.

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.

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [36]) {units=*NX_UNITLESS*}

Six equally formatted sextets chained together. For each sextett the first entry is an XDMF primitive topology key (here 5 for polygon), the second entry the XDMF primitive count value (here 4 because each face is a quad). The remaining four values are the vertex indices.

PROGRAM: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

visualization: (recommended) *NXprocess*

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [n_topology]) {units=*NX_UNITLESS*}

An array of triplets of integers which can serve as a supplementary array for Paraview to display the reconstructed dataset. The XDMF primitive type is here 1, the number of primitives 1 per triplet, the last integer in each triplet is the identifier of each point starting from zero.

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) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

DATA: (required) *NXdata*

A default three-dimensional histogram of the total number of ions in each bin obtained via using a rectangular transfer function.

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR* <=

data_counts: (required) *NX_NUMBER* (Rank: 3, Dimensions: [n_z, n_y, n_x]) {units=*NX_UNITLESS*} <=

Array of counts for each bin.

axis_z: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_z]) {units=*NX_LENGTH*}

Bin center of mass position along the z axis.

@long_name: (required) *NX_CHAR*

axis_y: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

Bin center of mass position along the y axis.

@long_name: (required) *NX_CHAR*

axis_x: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_x]) {units=*NX_LENGTH*}

Bin center of mass position along the x axis.

@long_name: (required) *NX_CHAR*

ranging: (recommended) *NXprocess*

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. Kühbach et al.

sequence_index: (recommended) *NX_POSINT <=*

number_of_ion_types: (required) *NX_UINT* {units=*NX_UNITLESS*}

How many ion types are distinguished. If no ranging was performed each ion is of the special unknown type. The iontype ID of this unknown type is 0 which is a reserve value. Consequently, iontypes start counting from 1.

maximum_number_of_atoms_per_molecular_ion: (required) *NX_UINT* {units=*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.

PROGRAM: (required) *NXprogram*

program: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

mass_to_charge_distribution: (recommended) *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. This (NXprocess) stores the settings of this binning. **range_minmax**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [2]) {units=*NX_ANY*}

Smallest and largest mass-to-charge-state ratio value.

range_increment: (required) *NX_FLOAT* {units=*NX_ANY*}

Binning width of the mass-to-charge-state ratio values.

PROGRAM: (required) *NXprogram*

program: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

mass_spectrum: (required) *NXdata*

A default histogram aka mass spectrum of the mass-to-charge-state ratio values. **@signal**: (required) *NX_CHAR <=*

@axes: (required) *NX_CHAR <=*

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR <=*

data_counts: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_bins]) {units=*NX_UNITLESS*} *<=*

Array of counts for each bin.

@long_name: (required) *NX_CHAR <=*

axis_mass_to_charge: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_bins]) {units=*NX_ANY*}

Right boundary of each mass-to-charge-state ratio value bin.

@long_name: (required) *NX_CHAR*

background_quantification: (recommended) *NXprocess*

Details of the background model which was used to correct the total counts per bin into counts. **PROGRAM**: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

peak_search_and_deconvolution: (recommended) *NXprocess*

How where peaks in the background-corrected in the histogram of mass-to-charge-state ratio values identified? **PROGRAM**: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

PEAK: (optional) *NXpeak*

label: (recommended) *NX_CHAR* <=

peak_model: (required) *NX_CHAR* <=

THIS DOCSTRING NEEDS CLARIFICATION.

peak_identification: (recommended) *NXprocess*

Details about how peaks, with taking into account error models, were interpreted as ion types or not. **PROGRAM**: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

ION: (optional) *NXion*

isotope_vector: (required) *NX_UINT* <=

charge_state: (required) *NX_INT* <=

mass_to_charge_range: (required) *NX_FLOAT* <=

nuclid_list: (recommended) *NX_UINT* <=

name: (recommended) *NX_CHAR* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm/ENTRY-group*
- */NXapm/ENTRY/atom_probe-group*
- */NXapm/ENTRY/atom_probe/analysis_chamber-group*
- */NXapm/ENTRY/atom_probe/analysis_chamber/CSG-group*
- */NXapm/ENTRY/atom_probe/analysis_chamber/description-field*
- */NXapm/ENTRY/atom_probe/analysis_chamber/FABRICATION-group*

- /NXapm/ENTRY/atom_probe/analysis_chamber/FABRICATION/capabilities-field
- /NXapm/ENTRY/atom_probe/analysis_chamber/FABRICATION/identifier-field
- /NXapm/ENTRY/atom_probe/analysis_chamber/FABRICATION/model-field
- /NXapm/ENTRY/atom_probe/analysis_chamber/FABRICATION/vendor-field
- /NXapm/ENTRY/atom_probe/analysis_chamber/name-field
- /NXapm/ENTRY/atom_probe/analysis_chamber/pressure-field
- /NXapm/ENTRY/atom_probe/buffer_chamber-group
- /NXapm/ENTRY/atom_probe/buffer_chamber/CSG-group
- /NXapm/ENTRY/atom_probe/buffer_chamber/description-field
- /NXapm/ENTRY/atom_probe/buffer_chamber/FABRICATION-group
- /NXapm/ENTRY/atom_probe/buffer_chamber/FABRICATION/capabilities-field
- /NXapm/ENTRY/atom_probe/buffer_chamber/FABRICATION/identifier-field
- /NXapm/ENTRY/atom_probe/buffer_chamber/FABRICATION/model-field
- /NXapm/ENTRY/atom_probe/buffer_chamber/FABRICATION/vendor-field
- /NXapm/ENTRY/atom_probe/buffer_chamber/name-field
- /NXapm/ENTRY/atom_probe/buffer_chamber/pressure-field
- /NXapm/ENTRY/atom_probe/control_software-group
- /NXapm/ENTRY/atom_probe/control_software/analysis_chamber-group
- /NXapm/ENTRY/atom_probe/control_software/buffer_chamber-group
- /NXapm/ENTRY/atom_probe/control_software/load_lock_chamber-group
- /NXapm/ENTRY/atom_probe/control_software/PROGRAM-group
- /NXapm/ENTRY/atom_probe/control_software/PROGRAM/program-field
- /NXapm/ENTRY/atom_probe/control_software/PROGRAM/program@version-attribute
- /NXapm/ENTRY/atom_probe/CSG-group
- /NXapm/ENTRY/atom_probe/FABRICATION-group
- /NXapm/ENTRY/atom_probe/FABRICATION/capabilities-field
- /NXapm/ENTRY/atom_probe/FABRICATION/identifier-field
- /NXapm/ENTRY/atom_probe/FABRICATION/model-field
- /NXapm/ENTRY/atom_probe/FABRICATION/vendor-field
- /NXapm/ENTRY/atom_probe/field_of_view-field
- /NXapm/ENTRY/atom_probe/flight_path_length-field
- /NXapm/ENTRY/atom_probe/getter_pump-group
- /NXapm/ENTRY/atom_probe/getter_pump/design-field
- /NXapm/ENTRY/atom_probe/getter_pump/FABRICATION-group
- /NXapm/ENTRY/atom_probe/getter_pump/FABRICATION/capabilities-field
- /NXapm/ENTRY/atom_probe/getter_pump/FABRICATION/CSG-group

- */NXapm/ENTRY/atom_probe/getter_pump/FABRICATION/identifier-field*
- */NXapm/ENTRY/atom_probe/getter_pump/FABRICATION/model-field*
- */NXapm/ENTRY/atom_probe/getter_pump/FABRICATION/vendor-field*
- */NXapm/ENTRY/atom_probe/hit_multiplicity-group*
- */NXapm/ENTRY/atom_probe/hit_multiplicity/hit_multiplicity-field*
- */NXapm/ENTRY/atom_probe/hit_multiplicity/PROGRAM-group*
- */NXapm/ENTRY/atom_probe/hit_multiplicity/PROGRAM/program-field*
- */NXapm/ENTRY/atom_probe/hit_multiplicity/PROGRAM/program@version-attribute*
- */NXapm/ENTRY/atom_probe/hit_multiplicity/pulse_id-field*
- */NXapm/ENTRY/atom_probe/hit_multiplicity/pulses_since_last_ion-field*
- */NXapm/ENTRY/atom_probe/hit_multiplicity/sequence_index-field*
- */NXapm/ENTRY/atom_probe/hit_quality_filtering-group*
- */NXapm/ENTRY/atom_probe/hit_quality_filtering/sequence_index-field*
- */NXapm/ENTRY/atom_probe/instrument_calibration-group*
- */NXapm/ENTRY/atom_probe/instrument_name-field*
- */NXapm/ENTRY/atom_probe/ion_detector-group*
- */NXapm/ENTRY/atom_probe/ion_detector/CSG-group*
- */NXapm/ENTRY/atom_probe/ion_detector/manufacturer_name-field*
- */NXapm/ENTRY/atom_probe/ion_detector/model-field*
- */NXapm/ENTRY/atom_probe/ion_detector/name-field*
- */NXapm/ENTRY/atom_probe/ion_detector/serial_number-field*
- */NXapm/ENTRY/atom_probe/ion_detector/signal_amplitude-field*
- */NXapm/ENTRY/atom_probe/ion_detector/type-field*
- */NXapm/ENTRY/atom_probe/ion_filtering-group*
- */NXapm/ENTRY/atom_probe/ion_filtering/evaporation_id_included-field*
- */NXapm/ENTRY/atom_probe/ion_filtering/PROGRAM-group*
- */NXapm/ENTRY/atom_probe/ion_filtering/PROGRAM/program-field*
- */NXapm/ENTRY/atom_probe/ion_filtering/PROGRAM/program@version-attribute*
- */NXapm/ENTRY/atom_probe/ion_filtering/sequence_index-field*
- */NXapm/ENTRY/atom_probe/ion_impact_positions-group*
- */NXapm/ENTRY/atom_probe/ion_impact_positions/arrival_time_pairs-field*
- */NXapm/ENTRY/atom_probe/ion_impact_positions/hit_positions-field*
- */NXapm/ENTRY/atom_probe/ion_impact_positions/PROGRAM-group*
- */NXapm/ENTRY/atom_probe/ion_impact_positions/PROGRAM/program-field*
- */NXapm/ENTRY/atom_probe/ion_impact_positions/PROGRAM/program@version-attribute*
- */NXapm/ENTRY/atom_probe/ion_impact_positions/sequence_index-field*

- */NXapm/ENTRY/atom_probe/load_lock_chamber-group*
- */NXapm/ENTRY/atom_probe/load_lock_chamber/CSG-group*
- */NXapm/ENTRY/atom_probe/load_lock_chamber/description-field*
- */NXapm/ENTRY/atom_probe/load_lock_chamber/FABRICATION-group*
- */NXapm/ENTRY/atom_probe/load_lock_chamber/FABRICATION/capabilities-field*
- */NXapm/ENTRY/atom_probe/load_lock_chamber/FABRICATION/identifier-field*
- */NXapm/ENTRY/atom_probe/load_lock_chamber/FABRICATION/model-field*
- */NXapm/ENTRY/atom_probe/load_lock_chamber/FABRICATION/vendor-field*
- */NXapm/ENTRY/atom_probe/load_lock_chamber/name-field*
- */NXapm/ENTRY/atom_probe/load_lock_chamber/pressure-field*
- */NXapm/ENTRY/atom_probe/local_electrode-group*
- */NXapm/ENTRY/atom_probe/local_electrode/APERTURE_EM-group*
- */NXapm/ENTRY/atom_probe/local_electrode/APERTURE_EM/FABRICATION-group*
- */NXapm/ENTRY/atom_probe/local_electrode/APERTURE_EM/FABRICATION/capabilities-field*
- */NXapm/ENTRY/atom_probe/local_electrode/APERTURE_EM/FABRICATION/identifier-field*
- */NXapm/ENTRY/atom_probe/local_electrode/APERTURE_EM/name-field*
- */NXapm/ENTRY/atom_probe/local_electrode/APERTURE_EM/value-field*
- */NXapm/ENTRY/atom_probe/local_electrode/CSG-group*
- */NXapm/ENTRY/atom_probe/local_electrode/name-field*
- */NXapm/ENTRY/atom_probe/location-field*
- */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-group*
- */NXapm/ENTRY/atom_probe/mass_to_charge_conversion/PROGRAM/program-field*
- */NXapm/ENTRY/atom_probe/mass_to_charge_conversion/PROGRAM/program@version-attribute*
- */NXapm/ENTRY/atom_probe/mass_to_charge_conversion/sequence_index-field*
- */NXapm/ENTRY/atom_probe/pulser-group*
- */NXapm/ENTRY/atom_probe/pulser/CSG-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/pulser/SOURCE-group*
- */NXapm/ENTRY/atom_probe/pulser/SOURCE/BEAM-group*
- */NXapm/ENTRY/atom_probe/pulser/SOURCE/BEAM/pinhole_position-group*

- */NXapm/ENTRY/atom_probe/pulser/SOURCE/BEAM/spot_position-group*
- */NXapm/ENTRY/atom_probe/pulser/SOURCE/FABRICATION-group*
- */NXapm/ENTRY/atom_probe/pulser/SOURCE/FABRICATION/capabilities-field*
- */NXapm/ENTRY/atom_probe/pulser/SOURCE/FABRICATION/identifier-field*
- */NXapm/ENTRY/atom_probe/pulser/SOURCE/FABRICATION/model-field*
- */NXapm/ENTRY/atom_probe/pulser/SOURCE/FABRICATION/vendor-field*
- */NXapm/ENTRY/atom_probe/pulser/SOURCE/name-field*
- */NXapm/ENTRY/atom_probe/pulser/SOURCE/power-field*
- */NXapm/ENTRY/atom_probe/pulser/SOURCE/pulse_energy-field*
- */NXapm/ENTRY/atom_probe/pulser/SOURCE/wavelength-field*
- */NXapm/ENTRY/atom_probe/pulser/standing_voltage-field*
- */NXapm/ENTRY/atom_probe/ranging-group*
- */NXapm/ENTRY/atom_probe/ranging/background_quantification-group*
- */NXapm/ENTRY/atom_probe/ranging/background_quantification/PROGRAM-group*
- */NXapm/ENTRY/atom_probe/ranging/background_quantification/PROGRAM/program-field*
- */NXapm/ENTRY/atom_probe/ranging/background_quantification/PROGRAM/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/axis_mass_to_charge-field*
- */NXapm/ENTRY/atom_probe/ranging/mass_to_charge_distribution/mass_spectrum/axis_mass_to_charge@long_name-attribute*
- */NXapm/ENTRY/atom_probe/ranging/mass_to_charge_distribution/mass_spectrum/data_counts-field*
- */NXapm/ENTRY/atom_probe/ranging/mass_to_charge_distribution/mass_spectrum/data_counts@long_name-attribute*
- */NXapm/ENTRY/atom_probe/ranging/mass_to_charge_distribution/mass_spectrum/title-field*
- */NXapm/ENTRY/atom_probe/ranging/mass_to_charge_distribution/mass_spectrum@axes-attribute*
- */NXapm/ENTRY/atom_probe/ranging/mass_to_charge_distribution/mass_spectrum@AXISNAME_indices-attribute*
- */NXapm/ENTRY/atom_probe/ranging/mass_to_charge_distribution/mass_spectrum@signal-attribute*
- */NXapm/ENTRY/atom_probe/ranging/mass_to_charge_distribution/PROGRAM-group*
- */NXapm/ENTRY/atom_probe/ranging/mass_to_charge_distribution/PROGRAM/program-field*
- */NXapm/ENTRY/atom_probe/ranging/mass_to_charge_distribution/PROGRAM/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/ION/name-field*
- */NXapm/ENTRY/atom_probe/ranging/peak_identification/ION/nuclid_list-field*
- */NXapm/ENTRY/atom_probe/ranging/peak_identification/PROGRAM-group*
- */NXapm/ENTRY/atom_probe/ranging/peak_identification/PROGRAM/program-field*
- */NXapm/ENTRY/atom_probe/ranging/peak_identification/PROGRAM/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/PEAK/label-field*
- */NXapm/ENTRY/atom_probe/ranging/peak_search_and_deconvolution/PEAK/peak_model-field*
- */NXapm/ENTRY/atom_probe/ranging/peak_search_and_deconvolution/PROGRAM-group*
- */NXapm/ENTRY/atom_probe/ranging/peak_search_and_deconvolution/PROGRAM/program-field*
- */NXapm/ENTRY/atom_probe/ranging/peak_search_and_deconvolution/PROGRAM/program@version-attribute*
- */NXapm/ENTRY/atom_probe/ranging/PROGRAM-group*
- */NXapm/ENTRY/atom_probe/ranging/PROGRAM/program-field*
- */NXapm/ENTRY/atom_probe/ranging/PROGRAM/program@version-attribute*
- */NXapm/ENTRY/atom_probe/ranging/sequence_index-field*
- */NXapm/ENTRY/atom_probe/reconstruction-group*
- */NXapm/ENTRY/atom_probe/reconstruction/crystallographic_calibration-field*
- */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/axis_x-field*
- */NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA/axis_x@long_name-attribute*
- */NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA/axis_y-field*
- */NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA/axis_y@long_name-attribute*
- */NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA/axis_z-field*
- */NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA/axis_z@long_name-attribute*
- */NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA/data_counts-field*
- */NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA/title-field*
- */NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA@axes-attribute*

- */NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA@AXISNAME_indices-attribute*
- */NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/DATA@signal-attribute*
- */NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/PROGRAM-group*
- */NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/PROGRAM/program-field*
- */NXapm/ENTRY/atom_probe/reconstruction/naive_point_cloud_density_map/PROGRAM/program@version-attribute*
- */NXapm/ENTRY/atom_probe/reconstruction/parameter-field*
- */NXapm/ENTRY/atom_probe/reconstruction/PROGRAM-group*
- */NXapm/ENTRY/atom_probe/reconstruction/PROGRAM/program-field*
- */NXapm/ENTRY/atom_probe/reconstruction/PROGRAM/program@version-attribute*
- */NXapm/ENTRY/atom_probe/reconstruction/protocol_name-field*
- */NXapm/ENTRY/atom_probe/reconstruction/reconstructed_positions-field*
- */NXapm/ENTRY/atom_probe/reconstruction/sequence_index-field*
- */NXapm/ENTRY/atom_probe/reconstruction/visualization-group*
- */NXapm/ENTRY/atom_probe/reconstruction/visualization/xdmf_topology-field*
- */NXapm/ENTRY/atom_probe/reconstruction/xdmf_topology-field*
- */NXapm/ENTRY/atom_probe/REFLECTRON-group*
- */NXapm/ENTRY/atom_probe/REFLECTRON/applied-field*
- */NXapm/ENTRY/atom_probe/REFLECTRON/CSG-group*
- */NXapm/ENTRY/atom_probe/REFLECTRON/description-field*
- */NXapm/ENTRY/atom_probe/REFLECTRON/FABRICATION-group*
- */NXapm/ENTRY/atom_probe/REFLECTRON/FABRICATION/capabilities-field*
- */NXapm/ENTRY/atom_probe/REFLECTRON/FABRICATION/identifier-field*
- */NXapm/ENTRY/atom_probe/REFLECTRON/FABRICATION/model-field*
- */NXapm/ENTRY/atom_probe/REFLECTRON/FABRICATION/vendor-field*
- */NXapm/ENTRY/atom_probe/REFLECTRON/name-field*
- */NXapm/ENTRY/atom_probe/roughening_pump-group*
- */NXapm/ENTRY/atom_probe/roughening_pump/design-field*
- */NXapm/ENTRY/atom_probe/roughening_pump/FABRICATION-group*
- */NXapm/ENTRY/atom_probe/roughening_pump/FABRICATION/capabilities-field*
- */NXapm/ENTRY/atom_probe/roughening_pump/FABRICATION/CSG-group*
- */NXapm/ENTRY/atom_probe/roughening_pump/FABRICATION/identifier-field*
- */NXapm/ENTRY/atom_probe/roughening_pump/FABRICATION/model-field*
- */NXapm/ENTRY/atom_probe/roughening_pump/FABRICATION/vendor-field*
- */NXapm/ENTRY/atom_probe/specimen_monitoring-group*
- */NXapm/ENTRY/atom_probe/specimen_monitoring/detection_rate-field*

- */NXapm/ENTRY/atom_probe/specimen_monitoring/estimated_field_at_the_apex-field*
- */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/stage_lab/CSG-group*
- */NXapm/ENTRY/atom_probe/stage_lab/temperature-field*
- */NXapm/ENTRY/atom_probe/status-field*
- */NXapm/ENTRY/atom_probe/turbomolecular_pump-group*
- */NXapm/ENTRY/atom_probe/turbomolecular_pump/design-field*
- */NXapm/ENTRY/atom_probe/turbomolecular_pump/FABRICATION-group*
- */NXapm/ENTRY/atom_probe/turbomolecular_pump/FABRICATION/capabilities-field*
- */NXapm/ENTRY/atom_probe/turbomolecular_pump/FABRICATION/CSG-group*
- */NXapm/ENTRY/atom_probe/turbomolecular_pump/FABRICATION/identifier-field*
- */NXapm/ENTRY/atom_probe/turbomolecular_pump/FABRICATION/model-field*
- */NXapm/ENTRY/atom_probe/turbomolecular_pump/FABRICATION/vendor-field*
- */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-group*
- */NXapm/ENTRY/atom_probe/voltage_and_bowl_correction/PROGRAM/program-field*
- */NXapm/ENTRY/atom_probe/voltage_and_bowl_correction/PROGRAM/program@version-attribute*
- */NXapm/ENTRY/atom_probe/voltage_and_bowl_correction/raw_tof-field*
- */NXapm/ENTRY/atom_probe/voltage_and_bowl_correction/sequence_index-field*
- */NXapm/ENTRY/atom_probe/voltage_and_bowl_correction/tof_calibration-group*
- */NXapm/ENTRY/COORDINATE_SYSTEM_SET-group*
- */NXapm/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-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/PROGRAM-group*
- */NXapm/ENTRY/PROGRAM/program-field*

- */NXapm/ENTRY/USER/social_media_platform-field*
- */NXapm/ENTRY/USER/telephone_number-field*
- */NXapm/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm.nxdl.xml

NXapm_composition_space_results

Status:

application definition, extends *NXobject*

Description:

Results of a run with Alaukik Saxena's composition space tool.

This is an initial draft application definition for the common NFDI-MatWerk, FAIRmat infrastructure use case IUC09 how to improve the organization and results storage of the composition space tool and make these data at the same time directly understandable for NOMAD.

This draft does no contain yet the annotations for how to also store in the HDF5 file a default visualization whereby the composition grid could directly be explored using H5Web. I am happy to add this ones the data have been mapped on this schema, i.e. more discussion needed.

Also iso-surfaces can be described, for paraprobe, this is a solved problem, check the respective group in the NXapm_paraprobe_results_nanochem data schema/application definition.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_voxel: Number of voxel of discretized domain for analyzed part of the dataset.

d: The dimensionality of the grid.

c: The cardinality or total number of cells/grid points.

n_clst_dict: Number of terms in the composition clustering dictionary

n_spat_dict: Number of terms in the position clustering dictionary

Groups cited:
NXapm_input_ranging, *NXapm_input_reconstruction*, *NXcg_grid*, *NXcoordinate_system_set*, *NXcs_computer*, *NXcs_cpu*, *NXcs_gpu*, *NXcs_io_obj*, *NXcs_io_sys*, *NXcs_mm_sys*, *NXcs_profiling_event*, *NXcs_profiling*, *NXentry*, *NXfabrication*, *NXion*, *NXprocess*, *NXprogram*, *NXtransformations*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_composition_space_results`

job_pyiron_identifier: (recommended) *NX_CHAR*

TBD, maybe how to link between pyiron state tracking and app state tracking

description: (optional) *NX_CHAR*

Disencouraged place for free-text for e.g. comments.

start_time: (required) *NX_DATE_TIME* <=

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file was started, i.e. when composition space tool was started as a process.

end_time: (required) *NX_DATE_TIME* <=

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file were completed and composition space tool exited as a process.

config_filename: (required) *NX_CHAR*

The path and name of the config file for this analysis. TBD, this can be e.g. Alaukik's YAML file for composition space. **@version**: (required) *NX_CHAR*

At least SHA256 strong hash of the specific config_file for tracking provenance.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

status: (required) *NX_CHAR*

A statement whether the executable managed to process the analysis or failed prematurely.

This status is written to the results file after the end_time at which point the executable must not compute any analysis. Only when this status message is present and shows *success*, the user should consider the results. In all other cases it might be that the executable has terminated prematurely or another error occurred.

Any of these values: `success` | `failure`

PROGRAM: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

dataset: (required) *NXapm_input_reconstruction*

filename: (required) *NX_CHAR* <=

The path and name of the file (technology partner or community format) from which reconstructed ion positions were loaded. **@version**: (required) *NX_CHAR* <=

iontypes: (required) *NXapm_input_ranging*

filename: (required) *NX_CHAR* <=

The path and name of the file (technology partner or community format from which ranging definitions, i.e. how to map mass-to- charge-state ratios on iontypes were loaded. **@version**: (required) *NX_CHAR* <=

USER: (recommended) *NXuser* <=

If used, contact information and eventually details of at least the person who performed this analysis. **name**: (required) *NX_CHAR* <=

affiliation: (recommended) *NX_CHAR* <=

address: (optional) *NX_CHAR* <=

email: (recommended) *NX_CHAR* <=

orcid: (recommended) *NX_CHAR* <=

orcid_platform: (recommended) *NX_CHAR* <=

telephone_number: (optional) *NX_CHAR* <=

role: (recommended) *NX_CHAR* <=

social_media_name: (optional) *NX_CHAR* <=

social_media_platform: (optional) *NX_CHAR* <=

COORDINATE_SYSTEM_SET: (optional) *NXcoordinate_system_set*

Details about the coordinate system conventions used.

TRANSFORMATIONS: (required) *NXtransformations* <=

The individual coordinate systems which should be used. Some suggestions follow, e.g. that field names should be prefixed with the following controlled terms indicating which individual coordinate system is described:

- world
- composition_space
- lab
- specimen
- laser
- leap
- detector
- recon

voxelization: (required) *NXprocess* <=

sequence_index: (required) *NX_POSINT* <=

Obligatory value: `1`

CG_GRID: (required) *NXcg_grid*

dimensionality: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

Any of these values: `1` | `2` | `3`

cardinality: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

origin: (required) *NX_NUMBER* (Rank: 1, Dimensions: [d]) {units=*NX_LENGTH*} <=

symmetry: (required) *NX_CHAR* <=

Obligatory value: `cubic`

cell_dimensions: (required) *NX_NUMBER* (Rank: 1, Dimensions: [d]) {units=*NX_LENGTH*} *<=*

extent: (required) *NX_POSINT* (Rank: 1, Dimensions: [d]) {units=*NX_UNITLESS*} *<=*

identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} *<=*

position: (required) *NX_NUMBER* (Rank: 2, Dimensions: [c, d]) {units=*NX_LENGTH*} *<=*

Position of each cell in Euclidean space.

coordinate: (optional) *NX_INT* (Rank: 2, Dimensions: [c, d]) {units=*NX_DIMENSIONLESS*} *<=*

voxel_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*}

For each ion, the identifier of the voxel in which the ion is located.

TRANSFORMATIONS: (recommended) *NXtransformations <=*

Reference to or definition of a coordinate system with which the positions and directions are interpretable.

ION: (optional) *NXion*

name: (required) *NX_CHAR <=*

composition: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_voxel])

clustering_composition_space: (required) *NXprocess <=*

In Alaukik's tool the GMM step.

sequence_index: (required) *NX_POSINT <=*

Obligatory value: `2`

cluster_dict_keyword: (required) *NX_CHAR* (Rank: 1, Dimensions: [n_clst_dict])

The keywords of the dictionary of distinguished compositionally- defined cluster, e.g. the phases. Examples for keywords could be phase1, phase2, and so one and so forth.

cluster_dict_value: (required) *NX_UINT* (Rank: 1, Dimensions: [n_clst_dict]) {units=*NX_UNITLESS*}

Resolves for each keyword in cluster_dict which integer is used to label something that it belongs or is assumed to represent this cluster.

cluster_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [n_voxel]) {units=*NX_UNITLESS*}

For example if the voxel grid is used to report that there are voxels which are assumed to represent volume of either phase1 or phase2, the cluster_dict_keyword would be a list with two names phase1 and phase2, respectively. The cluster_dict_value would be a list of e.g. integers 1 and 2. These could be used to build an array with as many entries as there are voxel and store in this array the respective value to encode which phase is assumed for each voxel.

clustering_real_space: (required) *NXprocess <=*

In Alaukik's tool the DBScan step after the GMM step.

sequence_index: (required) *NX_POSINT* <=

Obligatory value: 3

cluster_dict_keyword: (required) *NX_CHAR* (Rank: 1, Dimensions: [n_spat_dict])

The keywords of the dictionary of distinguished spatially-contiguous clusters. Examples for keywords could be precipitate1, precipitate2, and so one and so forth.

cluster_dict_value: (required) *NX_UINT* (Rank: 1, Dimensions: [n_spat_dict]) {units=*NX_UNITLESS*}

Resolves for each keyword in cluster_dict which integer is used to label something that it belongs or is assumed to represent this cluster.

cluster_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [n_voxel]) {units=*NX_UNITLESS*}

For example if the voxel grid is used to report that there are voxels which are assumed to represent volume of certain precipitates, say we found ten precipitates and consider the rest as matrix. We could make a list of say matrix, precipitate1, precipitate2, . . . , precipitate10. With cluster_dict_value then running from 0 to 10, i.e. matrix is flagged special as 0 and the remaining particles are indexed conveniently as 1, 2, . . . , 10 like end users expect.

performance: (required) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* <=

command_line_call: (optional) *NX_CHAR* <=

start_time: (recommended) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

total_elapsed_time: (required) *NX_NUMBER* <=

number_of_processes: (required) *NX_POSINT* <=

number_of_threads: (required) *NX_POSINT* <=

number_of_gpus: (required) *NX_POSINT* <=

CS_COMPUTER: (recommended) *NXcs_computer* <=

name: (recommended) *NX_CHAR* <=

operating_system: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

uuid: (optional) *NX_CHAR* <=

CS_CPU: (optional) *NXcs_cpu* <=

name: (optional) *NX_CHAR* <=

FABRICATION: (recommended) *NXfabrication* <=

identifier: (optional) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CS_GPU: (optional) *NXcs_gpu* <=

name: (optional) *NX_CHAR* <=

FABRICATION: (recommended) *NXfabrication* <=

identifier: (optional) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CS_MM_SYS: (optional) *NXcs_mm_sys* <=

total_physical_memory: (required) *NX_NUMBER* <=

CS_IO_SYS: (optional) *NXcs_io_sys* <=

CS_IO_OBJ: (required) *NXcs_io_obj* <=

technology: (required) *NX_CHAR* <=

max_physical_capacity: (required) *NX_NUMBER* <=

name: (optional) *NX_CHAR* <=

FABRICATION: (recommended) *NXfabrication* <=

identifier: (optional) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CS_PROFILING_EVENT: (required) *NXcs_profiling_event*

start_time: (optional) *NX_DATE_TIME* <=

end_time: (optional) *NX_DATE_TIME* <=

description: (required) *NX_CHAR* <=

elapsed_time: (required) *NX_NUMBER* <=

number_of_processes: (required) *NX_POSINT* <=

Specify if it was different from the number_of_processes in the NXcs_profiling super class.

number_of_threads: (required) *NX_POSINT* <=

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

number_of_gpus: (required) *NX_POSINT* <=

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

max_virtual_memory_snapshot: (recommended) *NX_NUMBER* <=

max_resident_memory_snapshot: (recommended) *NX_NUMBER* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_composition_space_results/ENTRY-group*
- */NXapm_composition_space_results/ENTRY/clustering_composition_space-group*
- */NXapm_composition_space_results/ENTRY/clustering_composition_space/cluster_dict_keyword-field*
- */NXapm_composition_space_results/ENTRY/clustering_composition_space/cluster_dict_value-field*
- */NXapm_composition_space_results/ENTRY/clustering_composition_space/cluster_identifier-field*
- */NXapm_composition_space_results/ENTRY/clustering_composition_space/sequence_index-field*
- */NXapm_composition_space_results/ENTRY/clustering_real_space-group*
- */NXapm_composition_space_results/ENTRY/clustering_real_space/cluster_dict_keyword-field*
- */NXapm_composition_space_results/ENTRY/clustering_real_space/cluster_dict_value-field*
- */NXapm_composition_space_results/ENTRY/clustering_real_space/cluster_identifier-field*
- */NXapm_composition_space_results/ENTRY/clustering_real_space/sequence_index-field*
- */NXapm_composition_space_results/ENTRY/config_filename-field*
- */NXapm_composition_space_results/ENTRY/config_filename@version-attribute*
- */NXapm_composition_space_results/ENTRY/COORDINATE_SYSTEM_SET-group*
- */NXapm_composition_space_results/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-group*
- */NXapm_composition_space_results/ENTRY/dataset-group*
- */NXapm_composition_space_results/ENTRY/dataset/filename-field*
- */NXapm_composition_space_results/ENTRY/dataset/filename@version-attribute*
- */NXapm_composition_space_results/ENTRY/definition-field*
- */NXapm_composition_space_results/ENTRY/description-field*
- */NXapm_composition_space_results/ENTRY/end_time-field*
- */NXapm_composition_space_results/ENTRY/iontypes-group*
- */NXapm_composition_space_results/ENTRY/iontypes/filename-field*
- */NXapm_composition_space_results/ENTRY/iontypes/filename@version-attribute*
- */NXapm_composition_space_results/ENTRY/job_pyiron_identifier-field*
- */NXapm_composition_space_results/ENTRY/performance-group*
- */NXapm_composition_space_results/ENTRY/performance/command_line_call-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER-group*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_CPU-group*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION-group*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/capabilities-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/identifier-field*

- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_CPU/name-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_GPU-group*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION-group*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/capabilities-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/identifier-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_GPU/name-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS-group*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ-group*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION-group*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/capabilitie*
  field
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identifier-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/max_physical_capacity-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/name-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/technology-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_MM_SYS-group*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_MM_SYS/total_physical_memory-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT-group*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/description-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/elapsed_time-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/end_time-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_resident_memory_sn*
  field
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_sna*
  field
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_gpus-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_processes-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_threads-field*

- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/start_time-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/name-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/operating_system-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/operating_system@version-attribute*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/uuid-field*
- */NXapm_composition_space_results/ENTRY/performance/current_working_directory-field*
- */NXapm_composition_space_results/ENTRY/performance/end_time-field*
- */NXapm_composition_space_results/ENTRY/performance/number_of_gpus-field*
- */NXapm_composition_space_results/ENTRY/performance/number_of_processes-field*
- */NXapm_composition_space_results/ENTRY/performance/number_of_threads-field*
- */NXapm_composition_space_results/ENTRY/performance/start_time-field*
- */NXapm_composition_space_results/ENTRY/performance/total_elapsed_time-field*
- */NXapm_composition_space_results/ENTRY/PROGRAM-group*
- */NXapm_composition_space_results/ENTRY/PROGRAM/program-field*
- */NXapm_composition_space_results/ENTRY/PROGRAM/program@version-attribute*
- */NXapm_composition_space_results/ENTRY/results_path-field*
- */NXapm_composition_space_results/ENTRY/start_time-field*
- */NXapm_composition_space_results/ENTRY/status-field*
- */NXapm_composition_space_results/ENTRY/USER-group*
- */NXapm_composition_space_results/ENTRY/USER/address-field*
- */NXapm_composition_space_results/ENTRY/USER/affiliation-field*
- */NXapm_composition_space_results/ENTRY/USER/email-field*
- */NXapm_composition_space_results/ENTRY/USER/name-field*
- */NXapm_composition_space_results/ENTRY/USER/orcid-field*
- */NXapm_composition_space_results/ENTRY/USER/orcid_platform-field*
- */NXapm_composition_space_results/ENTRY/USER/role-field*
- */NXapm_composition_space_results/ENTRY/USER/social_media_name-field*
- */NXapm_composition_space_results/ENTRY/USER/social_media_platform-field*
- */NXapm_composition_space_results/ENTRY/USER/telephone_number-field*
- */NXapm_composition_space_results/ENTRY/voxelization-group*
- */NXapm_composition_space_results/ENTRY/voxelization/CG_GRID-group*
- */NXapm_composition_space_results/ENTRY/voxelization/CG_GRID/cardinality-field*
- */NXapm_composition_space_results/ENTRY/voxelization/CG_GRID/cell_dimensions-field*
- */NXapm_composition_space_results/ENTRY/voxelization/CG_GRID/coordinate-field*
- */NXapm_composition_space_results/ENTRY/voxelization/CG_GRID/dimensionality-field*

- */NXapm_composition_space_results/ENTRY/voxelization/CG_GRID/extent-field*
- */NXapm_composition_space_results/ENTRY/voxelization/CG_GRID/identifier_offset-field*
- */NXapm_composition_space_results/ENTRY/voxelization/CG_GRID/origin-field*
- */NXapm_composition_space_results/ENTRY/voxelization/CG_GRID/position-field*
- */NXapm_composition_space_results/ENTRY/voxelization/CG_GRID/symmetry-field*
- */NXapm_composition_space_results/ENTRY/voxelization/CG_GRID/TRANSFORMATIONS-group*
- */NXapm_composition_space_results/ENTRY/voxelization/CG_GRID/voxel_identifier-field*
- */NXapm_composition_space_results/ENTRY/voxelization/ION-group*
- */NXapm_composition_space_results/ENTRY/voxelization/ION/composition-field*
- */NXapm_composition_space_results/ENTRY/voxelization/ION/name-field*
- */NXapm_composition_space_results/ENTRY/voxelization/sequence_index-field*
- */NXapm_composition_space_results/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_composition_space_results.nxdl.xml

NXapm_input_ranging

Status:

base class, extends *NXobject*

Description:

Metadata to ranging definitions made for a dataset in atom probe microscopy.

Ranging is the process of labeling time-of-flight data with so-called iontypes which ideally specify the most likely ion/molecular ion evaporated within a given mass-to-charge-state-ratio value interval.

The paraprobe-toolbox uses the convention that the so-called UNKNOWNTYPE iontype (or unranged ions) represents the default iontype. The ID of this special iontype is always reserved as 0. Each ion is assigned to the UNKNOWNTYPE by default. Iontypes are assigned by checking if the mass-to-charge-state-ratio values of an ion matches to any of the defined mass-to-charge-state-ratio intervals.

Symbols:

No symbol table

Groups cited:
none

Structure:

filename: (optional) *NX_CHAR*

Path and name of the NeXus/HDF5 file which stores ranging definitions.

@version: (optional) *NX_CHAR*

Version identifier of the file (representing an at least SHA256 strong) hash. Such hashes serve reproducibility as they can be used for tracking provenance metadata in a workflow.

group_name_iontypes: (optional) *NX_CHAR*

Name of the group (prefix to the individual ranging definitions) inside the file referred to by filename which points to the specific ranging definition to use. An HDF5 file can store multiple ranging definitions. Using an ID is the mechanism to distinguish which specific ranging (version) will be processed. Reconstruction and ranging IDs can differ. They specify different IDs.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_input_ranging/filename-field*
- */NXapm_input_ranging/filename@version-attribute*
- */NXapm_input_ranging/group_name_iontypes-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_input_ranging.nxdl.xml

NXapm_input_reconstruction

Status:

base class, extends *NXobject*

Description:

Metadata of a dataset (tomographic reconstruction) in atom probe microscopy.

Symbols:

No symbol table

Groups cited:
none

Structure:

filename: (optional) *NX_CHAR*

Name of the (NeXus)/HDF5 file which stores reconstructed ion position and mass-to-charge-state ratios. Such an HDF5 file can store multiple reconstructions. Using the information within the dataset_name fields is the mechanism whereby paraprobe decides which reconstruction to process. With this design it is possible that the same HDF5 file can store multiple versions of a reconstruction. **@version**: (optional) *NX_CHAR*

Version identifier of the file (representing an at least SHA256 strong) hash. Such hashes serve reproducibility as they can be used for tracking provenance metadata in a workflow.

dataset_name_reconstruction: (optional) *NX_CHAR*

Name of the dataset inside the HDF5 file which refers to the specific reconstructed ion positions to use for this analysis.

dataset_name_mass_to_charge: (optional) *NX_CHAR*

Name of the dataset inside the HDF5 file which refers to the specific mass-to-charge-state-ratio values to use for this analysis.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_input_reconstruction/dataset_name_mass_to_charge-field*
- */NXapm_input_reconstruction/dataset_name_reconstruction-field*
- */NXapm_input_reconstruction/filename-field*
- */NXapm_input_reconstruction/filename@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_input_reconstruction.nxdl.xml

NXapm_paraprobe_config_clusterer

Status:

application definition, extends *NXobject*

Description:

Configuration of a paraprobe-clusterer tool run in atom probe microscopy.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ivecmax: Maximum number of atoms per molecular ion. Should be 32 for paraprobe.

n_clust_algos: Number of clustering algorithms used.

n_ions: Number of different iontypes to distinguish during clustering.

Groups cited:
NXapm_input_ranging, *NXapm_input_reconstruction*, *NXcg_cylinder_set*, *NXcg_ellipsoid_set*, *NXcg_face_list_data_structure*, *NXcg_hexahedron_set*, *NXcs_filter_boolean_mask*, *NXentry*, *NXmatch_filter*, *NXprocess*, *NXspatial_filter*, *NXsubsampling_filter*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_config_clusterer`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

time_stamp: (required) *NX_DATE_TIME*

ISO 8601 formatted time code with local time zone offset to UTC information included when this configuration file was created.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

number_of_processes: (required) *NX_UINT* {units=*NX_UNITLESS*}

How many tasks to perform?

cameca_to_nexus: (optional) *NXprocess* <=

This process maps results from cluster analyses performed with IVAS/APSuite into an interoperable representation. Specifically in this process paraprobe-clusterer takes results from clustering methods from other tools of the APM community, like IVAS/APSuite. These results are usually reported in two ways. Either as an explicit list of reconstructed ion positions. In the case of IVAS these positions are reported through a text file with a cluster label for each position.

Alternatively, the list of positions is reported, as it is the case for AMETEK (IVAS/AP Suite) but the cluster labels are specified implicitly only in the following way: The mass-to-charge-state ratio column of a what is effectively a file formatted like POS is used to assign a hypothetical mass-to-charge value which resolves a floating point representation of the cluster ID.

Another case can occur where all disjoint floating point values, i.e. here cluster labels, are reported and then a dictionary is created how each value matches to a cluster ID.

In general the cluster ID zero is reserved for marking the dataset as to not be assigned to any cluster. Therefore, indices of disjoint clusters start at 1. **recover_evaporation_id**: (required) *NX_BOOLEAN*

Specifies if the tool should try to recover for each position the closest matching position from dataset/dataset_name_reconstruction (within floating point accuracy). This can be useful for instance when users wish to recover the original evaporation ID, which IVAS/AP Suite drops for instance when writing their *.indexed.* cluster results POS files.

dataset: (required) *NXapm_input_reconstruction*

filename: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

dataset_name_reconstruction: (required) *NX_CHAR* <=

dataset_name_mass_to_charge: (required) *NX_CHAR* <=

AMETEK/Cameca results of cluster analyses, like with the maximum-separation (MS) method clustering algorithm J. Hyde et al. are stored as an improper POS file: This is a matrix of floating point quadruplets, one for each ion and as many quadruplets as ions were investigated. The first three values encode the position of the ion. The fourth value is an improper mass-to-charge-state-ratio value which encodes the integer identifier of the cluster as a floating point number.

cluster_analysis: (optional) *NXprocess* <=

This process performs a cluster analysis on a reconstructed dataset or a portion of the reconstruction. **ion_type_filter**: (required) *NX_CHAR*

How should iontypes be interpreted/considered during the cluster analysis. Different options exist how iontypes are interpreted (if considered at all) given an iontype represents in general a (molecular) ion with different isotopes that have individually different multiplicity.

The value resolve_all will set an ion active in the analysis regardless of which iontype it is. The value resolve_unknown will set an ion active when it is of the UNKNOWNTYPE. The value resolve_ion will set an ion active if it is of the specific iontype, irregardless of its elemental or isotopic details. The value resolve_element will set an ion active, and most importantly, account as many times for it, as the (molecular) ion contains atoms of elements in the whitelist ion_query_isotope_vector. The value resolve_isotope will set an ion active, and most importantly, account as many times for it, as the (molecular) ion contains isotopes in the whitelist ion_query_isotope_vector.

In effect, ion_query_isotope_vector acts as a whitelist to filter which ions are considered as source ions of the correlation statistics and how the multiplicity of each ion will be factorized.

This is relevant as in atom probe we have the situation that a ion of a molecular ion with more than one nuclid, say Ti O for example is counted such that although there is a single TiO molecular ion at a position that the cluster has two members. This multiplicity affects the size of the feature and chemical composition.

Obligatory value: `resolve_element`

ion_query_isotope_vector: (required) *NX_UINT* (Rank: 2, Dimensions: [n_ions, n_ivecmax]) {units=*NX_UNITLESS*}

Matrix of isotope vectors, as many as rows as different candidates for iontypes should be distinguished as possible source iontypes. In the simplest case, the matrix contains only the proton number of the element in the row, all other values set to zero. Combined with ion_query_type_source set to resolve_element this will recover usual spatial correlation statistics like the 1NN C-C spatial statistics.

dataset: (required) *NXapm_input_reconstruction*

filename: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

dataset_name_reconstruction: (required) *NX_CHAR <=*

dataset_name_mass_to_charge: (required) *NX_CHAR <=*

iontypes: (required) *NXapm_input_ranging*

filename: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

group_name_iontypes: (required) *NX_CHAR <=*

ion_to_edge_distances: (optional) *NXprocess*

The tool enables to inject precomputed distance information for each point/ion which can be used for further post-processing and analysis. **filename**: (required) *NX_CHAR*

Name of an HDF5 file which contains the ion distances.

@version: (required) *NX_CHAR*

Version identifier of the file such as a secure hash which documents the binary state of the file to add an additional layer of reproducibility from which file specifically contains these data.

dataset_name: (required) *NX_CHAR*

Absolute HDF5 path to the dataset with distance values for each ion.

spatial_filter: (optional) *NXspatial_filter*

windowing_method: (required) *NX_CHAR <=*

Obligatory value: `entire_dataset`

CG_ELLIPSOID_SET: (optional) *NXcg_ellipsoid_set <=*

dimensionality: (required) *NX_POSINT <=*

cardinality: (required) *NX_POSINT <=*

identifier_offset: (required) *NX_INT <=*

center: (required) *NX_NUMBER <=*

half_axes_radii: (required) *NX_NUMBER <=*

orientation: (required) *NX_NUMBER <=*

CG_CYLINDER_SET: (optional) *NXcg_cylinder_set <=*

dimensionality: (required) *NX_POSINT <=*

cardinality: (required) *NX_POSINT <=*

identifier_offset: (required) *NX_INT <=*

center: (required) *NX_NUMBER <=*

height: (required) *NX_NUMBER <=*

radii: (required) *NX_NUMBER <=*

CG_HEXAHEDRON_SET: (optional) *NXcg_hexahedron_set <=*

dimensionality: (required) *NX_POSINT <=*

cardinality: (required) *NX_POSINT <=*

identifier_offset: (required) *NX_INT <=*

hexahedra: (required) *NXcg_face_list_data_structure <=*

CS_FILTER_BOOLEAN_MASK: (optional) *NXcs_filter_boolean_mask <=*

number_of_objects: (required) *NX_UINT <=*

bitdepth: (required) *NX_UINT <=*

mask: (required) *NX_UINT <=*

identifier: (required) *NX_UINT <=*

evaporation_id_filter: (optional) *NXsubsampling_filter*

iontype_filter: (optional) *NXmatch_filter*

hit_multiplicity_filter: (optional) *NXmatch_filter*

dbscan: (required) *NXprocess*

Settings for DBScan clustering algorithm. For original details about the algorithms and (performance-relevant) details consider:

- M. Ester et al.
- M. Götz et al.

For details about how the DBScan algorithms is the key behind the specific modification known as the maximum-separation method in the atom probe community consider E. Jägle et al. **high_throughput_method**: (required) *NX_CHAR*

Strategy how runs are performed with different parameter:

- For tuple as many runs are performed as parameter values.
- For combinatorics individual parameter arrays are looped over.

As an example we may define eps with ten entries and min_pts with three entries. If high_throughput_method is tuple the analysis is invalid as we have an insufficient number of min_pts for the ten eps values. By contrast, for combinatorics paraprobe-clusterer will run three individual min_pts runs for each eps value, resulting in a total of 30 analyses. As an example the DBScan analysis reported in M. Kühbach et al. would have defined an array of values np.linspace(0.2, 5.0, nums=241, endpoint=True) eps values, min_pts one, and high_throughput_method set to combinatorics.

Any of these values: `tuple` | `combinatorics`

eps: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_LENGTH*}

Array of epsilon (eps) parameter values.

min_pts: (required) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

Array of minimum points (min_pts) parameter values.

optics: (required) *NXprocess*

Settings for the OPTICS clustering algorithm.

- M. Ankerest et al.

high_throughput_method: (required) *NX_CHAR*

Strategy how runs are performed with different parameter:

- For tuple as many runs are performed as parameter values.
- For combinatorics individual parameter arrays are looped over.

See the explanation for the corresponding parameter for dbscan processes above-mentioned for further details.

Any of these values: `tuple` | `combinatorics`

min_pts: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Array of minimum points (min_pts) parameter values.

max_eps: (required) *NX_FLOAT* (Rank: 1, Dimensions: [j]) {units=*NX_LENGTH*}

Array of maximum epsilon (eps) parameter values.

hdbscan: (required) *NXprocess*

Settings for the HPDBScan clustering algorithm.

- L. McInnes et al. <https://dx.doi.org/10.21105/joss.00205>`_
- scikit-learn hdbscan library https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

See also this documentation for details about the parameter. Here we use the terminology of the hdbscan documentation. **high_throughput_method**: (required) *NX_CHAR*

Strategy how runs are performed with different parameter:

- For tuple as many runs are performed as parameter values.
- For combinatorics individual parameter arrays are looped over.

See the explanation for the corresponding parameter for dbscan processes above-mentioned for further details.

Any of these values: `tuple` | `combinatorics`

min_cluster_size: (required) *NX_NUMBER* (Rank: 1, Dimensions: [i]) {units=*NX_ANY*}

Array of min_cluster_size parameter values.

min_samples: (required) *NX_NUMBER* (Rank: 1, Dimensions: [j]) {units=*NX_ANY*}

Array of min_samples parameter values.

cluster_selection_epsilon: (required) *NX_NUMBER* (Rank: 1, Dimensions: [k]) {units=*NX_ANY*}

Array of cluster_selection parameter values.

alpha: (required) *NX_NUMBER* (Rank: 1, Dimensions: [m]) {units=*NX_ANY*}

Array of alpha parameter values.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_config_clusterer/ENTRY-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/analysis_description-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cameca_to_nexus-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cameca_to_nexus/dataset-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cameca_to_nexus/dataset/dataset_name_mass_to_charge-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cameca_to_nexus/dataset/dataset_name_reconstruction-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cameca_to_nexus/dataset/filename-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cameca_to_nexus/dataset/filename@version-attribute*
- */NXapm_paraprobe_config_clusterer/ENTRY/cameca_to_nexus/recover_evaporation_id-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/dataset-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/dataset/dataset_name_mass_to_charge-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/dataset/dataset_name_reconstruction-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/dataset/filename-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/dataset/filename@version-attribute*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/dbscan-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/dbscan/eps-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/dbscan/high_throughput_method-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/dbscan/min_pts-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/evaporation_id_filter-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/hdbscan-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/hdbscan/alpha-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/hdbscan/cluster_selection_epsilon-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/hdbscan/high_throughput_method-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/hdbscan/min_cluster_size-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/hdbscan/min_samples-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/hit_multiplicity_filter-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/ion_query_isotope_vector-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/ion_to_edge_distances-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/ion_to_edge_distances/dataset_name-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/ion_to_edge_distances/filename-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/ion_to_edge_distances/filename@version-attribute*

- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/ion_type_filter-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/iontype_filter-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/iontypes-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/iontypes/filename-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/iontypes/filename@version-attribute*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/iontypes/group_name_iontypes-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/optics-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/optics/high_throughput_method-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/optics/max_eps-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/optics/min_pts-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_CYLINDER_SET-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_CYLINDER_SET/cardinality-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_CYLINDER_SET/center-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_CYLINDER_SET/dimensionality-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_CYLINDER_SET/height-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_CYLINDER_SET/identifier_offset-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_CYLINDER_SET/radii-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_ELLIPSOID_SET-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_ELLIPSOID_SET/cardinality-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_ELLIPSOID_SET/center-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_ELLIPSOID_SET/dimensionality-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_ELLIPSOID_SET/half_axes_radii-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_ELLIPSOID_SET/identifier_offset-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_ELLIPSOID_SET/orientation-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_HEXAHEDRON_SET-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_HEXAHEDRON_SET/cardinality-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_HEXAHEDRON_SET/dimensionality-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_HEXAHEDRON_SET/hexahedra-group*

- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CG_HEXAHEDRON_SET/identifier_offset-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CS_FILTER_BOOLEAN_MASK-group*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CS_FILTER_BOOLEAN_MASK/bitdepth-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CS_FILTER_BOOLEAN_MASK/identifier-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CS_FILTER_BOOLEAN_MASK/mask-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/CS_FILTER_BOOLEAN_MASK/number_of_objects-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/cluster_analysis/spatial_filter/windowing_method-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/definition-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/number_of_processes-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/program-field*
- */NXapm_paraprobe_config_clusterer/ENTRY/program@version-attribute*
- */NXapm_paraprobe_config_clusterer/ENTRY/time_stamp-field*
- */NXapm_paraprobe_config_clusterer/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_config_clusterer.nxdl.xml

NXapm_paraprobe_config_distancer

Status:

application definition, extends *NXobject*

Description:

Configuration/settings of a paraprobe-distancer software tool run.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXapm_input_ranging, *NXapm_input_reconstruction*, *NXcg_cylinder_set*, *NXcg_ellipsoid_set*, *NXcg_face_list_data_structure*, *NXcg_hexahedron_set*, *NXcs_filter_boolean_mask*, *NXcs_profiling*, *NXentry*, *NXmatch_filter*, *NXprocess*, *NXspatial_filter*, *NXsubsampling_filter*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_config_distancer`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

time_stamp: (required) *NX_DATE_TIME*

ISO 8601 formatted time code with local time zone offset to UTC information included when this configuration file was created.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

number_of_processes: (required) *NX_UINT* {units=*NX_UNITLESS*}

How many individual analyses should the tool execute.

PROCESS: (required) *NXprocess* *<=*

dataset: (required) *NXapm_input_reconstruction*

filename: (required) *NX_CHAR* *<=*

@version: (required) *NX_CHAR* *<=*

dataset_name_reconstruction: (required) *NX_CHAR* *<=*

dataset_name_mass_to_charge: (required) *NX_CHAR* *<=*

iontypes: (required) *NXapm_input_ranging*

filename: (required) *NX_CHAR* *<=*

@version: (required) *NX_CHAR* *<=*

group_name_iontypes: (required) *NX_CHAR* *<=*

spatial_filter: (optional) *NXspatial_filter*

windowing_method: (required) *NX_CHAR* *<=*

CG_ELLIPSOID_SET: (optional) *NXcg_ellipsoid_set* *<=*

dimensionality: (required) *NX_POSINT* *<=*

cardinality: (required) *NX_POSINT* *<=*

identifier_offset: (required) *NX_INT* *<=*

center: (required) *NX_NUMBER <=*

half_axes_radii: (required) *NX_NUMBER <=*

orientation: (required) *NX_NUMBER <=*

CG_CYLINDER_SET: (optional) *NXcg_cylinder_set <=*

dimensionality: (required) *NX_POSINT <=*

cardinality: (required) *NX_POSINT <=*

identifier_offset: (required) *NX_INT <=*

center: (required) *NX_NUMBER <=*

height: (required) *NX_NUMBER <=*

radii: (required) *NX_NUMBER <=*

CG_HEXAHEDRON_SET: (optional) *NXcg_hexahedron_set <=*

dimensionality: (required) *NX_POSINT <=*

cardinality: (required) *NX_POSINT <=*

identifier_offset: (required) *NX_INT <=*

hexahedra: (required) *NXcg_face_list_data_structure <=*

CS_FILTER_BOOLEAN_MASK: (optional) *NXcs_filter_boolean_mask <=*

number_of_objects: (required) *NX_UINT <=*

bitdepth: (required) *NX_UINT <=*

mask: (required) *NX_UINT <=*

identifier: (required) *NX_UINT <=*

evaporation_id_filter: (optional) *NXsubsampling_filter*

iontype_filter: (optional) *NXmatch_filter*

hit_multiplicity_filter: (optional) *NXmatch_filter*

point_to_triangle: (required) *NXprocess*

Compute for all filtered points, e.g. ions of the point set the shortest Euclidean distance to the closest triangle of the set of triangles. The triangles can formed a closed surface mesh. Distances are not simple distances based on normal projections but giving an exact solution. **method**: (required) *NX_CHAR*

Specifies for which ions/points the tool will compute distances. The purpose of this setting is to avoid unnecessary computations when the user requests to only compute distances of ions within a threshold distance to the triangle soup.

By default the distances are computed for all ions; however the setting skin enables to compute distances only for those ions which are not farther away located to a triangle than threshold_distance.

Any of these values: `default` | `skin`

threshold_distance: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Maximum distance for which distances are computed when method is skin.

triangle_soup: (required) *NXprocess*

Paraprobe-distancer enables the computation of the Euclidean shortest distance for each member of a set of points against a set of triangles. In contrast to comparable methods used in atom probe the here computed distance is not simply the projected distance to one of the triangles but the more costly but robust computation of the distance between a point and a triangle.

The triangles can represent for instance the facets of a triangulated surface mesh of a model for the edge of the dataset. Such a model can be computed with paraprobe-surfacer. Alternatively, the triangles can be those from the set of all facets for a set of convex hulls, alpha-shapes, or alpha wrappings about three-dimensional objects like precipitates (computed with e.g. paraprobe-nanochem).

Currently, the tool does not check if the respectively specified triangle sets are consistent, what their topology is, or whether or not they are consistently oriented. Each dataset that is referred to in the list_of _dataset_names_vertices should be an (Nvertices, 3) array of NX_FLOAT. Each dataset referred to in the list_of_dataset_names_facet_indices should be an (Nfacets, 3) array of NX_UINT. Facet indices refer to vertex indices. These need to start at zero and must not exceed Nvertices - 1, i.e. the identifier_offset is 0 and vertices are indexed thus implicitly. Facet normal vectors have to be also an array of shape (Nfacets, 3) of NX_FLOAT. **number_of_files**: (required) *NX_UINT* {units=*NX_UNITLESS*}

How many triangle sets to consider.

PROCESS: (required) *NXprocess*

List of triangle sets. This design allows users to combine multiple triangle sets. **filename**: (required) *NX_CHAR*

Name of the HDF5 file(s) which contain(s) vertex coordinates and facet indices to describe the desired set of triangles. **@version**: (required) *NX_CHAR*

Version identifier of the file such as a secure hash which documents the binary state of the file to add an additional layer of reproducibility.

dataset_name_vertices: (required) *NX_CHAR*

Absolute HDF5 path to the dataset which specifies the array of vertex positions.

dataset_name_facet_indices: (required) *NX_CHAR*

Absolute HDF5 path to the dataset which specifies the array of facet indices.

dataset_name_facet_normals: (optional) *NX_CHAR*

Absolute HDF5 path to the dataset which specifies the array of facet normal vectors.

performance: (required) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR <=*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_config_distancer/ENTRY-group*
- */NXapm_paraprobe_config_distancer/ENTRY/analysis_description-field*
- */NXapm_paraprobe_config_distancer/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_config_distancer/ENTRY/definition-field*
- */NXapm_paraprobe_config_distancer/ENTRY/number_of_processes-field*
- */NXapm_paraprobe_config_distancer/ENTRY/performance-group*
- */NXapm_paraprobe_config_distancer/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/dataset-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/dataset/dataset_name_mass_to_charge-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/dataset/dataset_name_reconstruction-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/dataset/filename-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/dataset/filename@version-attribute*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/evaporation_id_filter-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/hit_multiplicity_filter-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/iontype_filter-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/iontypes-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/iontypes/filename-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/iontypes/filename@version-attribute*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/iontypes/group_name_iontypes-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/point_to_triangle-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/point_to_triangle/method-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/point_to_triangle/threshold_distance-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/point_to_triangle/triangle_soup-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/point_to_triangle/triangle_soup/number_of_files-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/point_to_triangle/triangle_soup/PROCESS-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/point_to_triangle/triangle_soup/PROCESS/dataset_name_facet_indices field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/point_to_triangle/triangle_soup/PROCESS/dataset_name_facet_norma field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/point_to_triangle/triangle_soup/PROCESS/dataset_name_vertices-field*

- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/point_to_triangle/triangle_soup/PROCESS/filename-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/point_to_triangle/triangle_soup/PROCESS/filename@version-attribute*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/cardinality-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/center-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/dimensionality-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/height-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/identifier_offset-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/radii-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/cardinality-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/center-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/dimensionality-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/half_axes_radii-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/identifier_offset-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/orientation-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/cardinality-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/dimensionality-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/hexahedra-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/identifier_offset-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK-group*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/bitdepth-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/identifier-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/mask-field*

- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/number_of_objects-field*
- */NXapm_paraprobe_config_distancer/ENTRY/PROCESS/spatial_filter/windowing_method-field*
- */NXapm_paraprobe_config_distancer/ENTRY/program-field*
- */NXapm_paraprobe_config_distancer/ENTRY/program@version-attribute*
- */NXapm_paraprobe_config_distancer/ENTRY/results_path-field*
- */NXapm_paraprobe_config_distancer/ENTRY/time_stamp-field*
- */NXapm_paraprobe_config_distancer/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_config_distancer.nxdl.xml

NXapm_paraprobe_config_intersector

Status:

application definition, extends *NXobject*

Description:

Configuration of a paraprobe-intersector tool run in atom probe microscopy.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXcollection, *NXcs_profiling*, *NXentry*, *NXprocess*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_config_intersector`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

time_stamp: (required) *NX_DATE_TIME*

ISO 8601 formatted time code with local time zone offset to UTC information included when this configuration file was created.

number_of_processes: (required) *NX_UINT* {units=*NX_UNITLESS*}

For now a support field for the tool to identify how many individual analyses the tool should execute as part of the analysis.

VOLUME_VOLUME_SPATIAL_CORRELATION: (required) *NXprocess* <=

Tracking volume_volume_spatial_correlation is the process of building logical relations between volumetric features based on meshes, their proximity and eventual intersections. Volumetric overlap and proximity of volumetric features is identified for members of sets of features to members of other sets of volumetric features. Specifically, for each time step k pairs of sets are compared: Members of a so-called current_set to members of a so-called next_set. Members can be different types of volumetric features. In the analysis of M. Kuehbach et al. specifically features can be so-called objects (closed non-degnerated polyhedra representing watertight parts of an e.g. iso-surface) and/or proxies. Proxies are computed doppelganger/replacement meshes for parts of an iso-surface which initially were not resulting in watertight meshes because objects at the edge of the dataset or incompletely measured or truncated objects. **intersection_detection_method**: (required) *NX_CHAR*

Specifies the method whereby to decide if two objects intersect volumetrically. For reasons which are detailed in the supplementary material of M. Kühbach et al., the tool by default assumes that two objects intersect if they share at least one ion with the same evaporation ID (shared_ion). Alternatively, with specifying tetrahedra_intersections, the tool can perform an intersection analysis which attempts to tetrahedralize first each polyhedron. If successful, the tool then checks for at least one pair of intersecting tetrahedra to identify if two objects intersect or not.

However, we found that these geometrical analyses can result in corner cases which the currently used library (TetGen) was not unable to tetrahedralize successfully. These cases were virtually always associated with complicated non-convex polyhedra which had portions of the mesh that were connected by almost point like tubes of triangles. Finding more robust methods for computing intersections between not necessarily convex polyhedra might improve the situation in the future.

Obligatory value: `shared_ion`

analyze_intersection: (required) *NX_BOOLEAN*

Specifies if the tool evaluates if for each pair the two objects (and proxies if used) intersect volumetrically.

analyze_proximity: (required) *NX_BOOLEAN*

Specifies if the tool evaluates if for each pair the two objects (and proxies if used) lie closer to one another than the threshold_proximity.

analyze_coprecipitation: (required) *NX_BOOLEAN*

Specifies if the tool evaluates, ones all tracking tasks were successfully completed, how intersecting or proximity related objects build sub-graphs. This is the feature which enabled M. Kühbach et al. 2022 the high-throughput analyses of how many objects are coprecipitates in the sense that they are single, duplet, triplet, or high-order. For these analyses to work has_object_volume needs to be activated.

threshold_proximity: (required) *NX_FLOAT* {units=*NX_LENGTH*}

The maximum Euclidean distance between two objects below which both objects are still considered within proximity.

has_current_to_next_links: (required) *NX_BOOLEAN*

Specifies if the tool stores the so-called forward relations between nodes representing members of the current_set to nodes representing members of the next_set.

has_next_to_current_links: (required) *NX_BOOLEAN*

Specifies if the tool stores the so-called backward relations between nodes representing members of the next_set to nodes representing members of the current_set.

current_set: (required) *NXprocess*

Current set stores a set of members, meshes of volumetric features, which will be checked for proximity and/or volumetric intersection, to members of the current_set. The meshes were generated as a result of some other meshing process. **set_identifier**: (required) *NX_UINT* {units=*NX_ANY*}

This identifier can be used to label the current set. The label effectively represents (can be interpreted as) the time/iteration step when the current set was taken. As it is detailed in M. Kühbach et al. 2022, this identifier takes the role of the time variable k.

number_of_feature_types: (required) *NX_UINT* {units=*NX_UNITLESS*}

The total number of distinguished feature sets FEATURE. It is assumed that the members within all these FEATURE sets are representing a set together. As an example this set might represent all volumetric_features. However, users might have formed a subset of this set where individuals were regrouped. For paraprobe-nanochem this is the case for objects and proxies. Specifically, objects are distinguished further into those far from and those close to the edge of the dataset. Similarly, proxies are distinguished further into those far from and those close to the edge of the dataset. So while these four sub-sets contain different so-called types of features key is that they were all generated for one set, here the current_set.

FEATURE: (required) *NXcollection*

feature_type: (required) *NX_CHAR*

Descriptive category explaining what these features are.

Any of these values:

- `objects_far_from_edge`
- `objects_close_to_edge`
- `proxies_far_from_edge`

- `proxies_close_to_edge`

filename: (required) *NX_CHAR*

Name of the (NeXus)/HDF5 file which contains triangulated surface meshes of the members of the set as instances of NXcg_polyhedron_set. **@version**: (required) *NX_CHAR*

Version identifier of the file such as a secure hash which documents the binary state of the file to add an additional layer of reproducibility from which file specifically contains these data.

groupname_geometry_prefix: (required) *NX_CHAR*

String whereby the path to the geometry data can be interferred automatically. Currently groupname_geometry_prefix/object<ID>/polyhedron.

feature_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Array of identifier whereby the path to the geometry data can be interferred automatically.

next_set: (required) *NXcollection*

Next set stores a set of members, meshes of volumetric features, which will be checked for proximity and/or volumetric intersection, to members of the next_set. The meshes were generated as a result of some other meshing process. **set_identifier**: (required) *NX_UINT* {units=*NX_ANY*}

This identifier can be used to label the next_set. The label effectively represents (can be interpreted as) the time/iteration step when the current set was taken. As it is detailed in M. Kühbach et al. 2022, this identifier takes the role of the time variable $k + 1$.

number_of_feature_types: (required) *NX_UINT* {units=*NX_UNITLESS*}

The total number of distinguished feature sets FEATURE. It is assumed that the members within all these FEATURE sets are representing a set together. As an example this set might represent all volumetric_features. However, users might have formed a subset of this set where individuals were regrouped. For paraprobe-nanochem this is the case for objects and proxies. Specifically, objects are distinguished further into those far from and those close to the edge of the dataset. Similarly, proxies are distinguished further into those far from and those close to the edge of the dataset. So while these four sub-sets contain different so-called types of features key is that they were all generated for one set, here the next_set.

FEATURE: (required) *NXcollection*

feature_type: (required) *NX_CHAR*

Descriptive category explaining what these features are.

Any of these values:

- `objects_far_from_edge`
- `objects_close_to_edge`

- `proxies_far_from_edge`
- `proxies_close_to_edge`

filename: (required) *NX_CHAR*

Name of the (NeXus)/HDF5 file which contains triangulated surface meshes of the members of the set as instances of NXcg_polyhedron_set. **@version**: (required) *NX_CHAR*

Version identifier of the file such as a secure hash which documents the binary state of the file to add an additional layer of reproducibility from which file specifically contains these data.

groupname_geometry_prefix: (required) *NX_CHAR*

String whereby the path to the geometry data can be interferred automatically. Currently groupname_geometry_prefix/object<ID>/polyhedron.

feature_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

Array of identifier whereby the path to the geometry data can be interferred automatically.

performance: (required) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* *<=*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_config_intersector/ENTRY-group*
- */NXapm_paraprobe_config_intersector/ENTRY/analysis_description-field*
- */NXapm_paraprobe_config_intersector/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_config_intersector/ENTRY/definition-field*
- */NXapm_paraprobe_config_intersector/ENTRY/number_of_processes-field*
- */NXapm_paraprobe_config_intersector/ENTRY/performance-group*
- */NXapm_paraprobe_config_intersector/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_config_intersector/ENTRY/program-field*
- */NXapm_paraprobe_config_intersector/ENTRY/program@version-attribute*
- */NXapm_paraprobe_config_intersector/ENTRY/results_path-field*
- */NXapm_paraprobe_config_intersector/ENTRY/time_stamp-field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION-group*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/analyze_coprecipitation-field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/analyze_intersection-field*

- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/analyze_proximity-field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/current_set-group*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/current_set/FEATURE-group*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/current_set/FEATURE/feature_ field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/current_set/FEATURE/feature_ field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/current_set/FEATURE/filenam field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/current_set/FEATURE/filenam attribute*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/current_set/FEATURE/groupn field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/current_set/number_of_feature field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/current_set/set_identifier-field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/has_current_to_next_links-field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/has_next_to_current_links-field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/intersection_detection_method field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/next_set-group*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/next_set/FEATURE-group*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/next_set/FEATURE/feature_id field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/next_set/FEATURE/feature_typ field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/next_set/FEATURE/filename-field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/next_set/FEATURE/filename@ attribute*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/next_set/FEATURE/groupnam field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/next_set/number_of_feature_ty field*

- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/next_set/set_identifier-field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/threshold_proximity-field*
- */NXapm_paraprobe_config_intersector/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_config_intersector.nxdl.xml

NXapm_paraprobe_config_nanochem

Status:

application definition, extends *NXobject*

Description:

Configuration of a paraprobe-nanochem tool run in atom probe microscopy.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ityp_deloc_cand: How many iontypes does the delocalization filter specify.

n_control_pts: How many disjoint control points are defined.

n_fct_filter_cand: How many iontypes does the interface meshing iontype filter specify.

n_fct_iterations: How many DCOM iterations.

n_ivec: Maximum number of atoms per molecular ion.

Groups cited:
NXapm_input_ranging, *NXapm_input_reconstruction*, *NXcg_cylinder_set*, *NXcg_ellipsoid_set*, *NXcg_face_list_data_structure*, *NXcg_hexahedron_set*, *NXcs_filter_boolean_mask*, *NXcs_profiling*, *NXentry*, *NXmatch_filter*, *NXprocess*, *NXspatial_filter*, *NXsubsampling_filter*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_config_nanochem`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

time_stamp: (required) *NX_DATE_TIME*

ISO 8601 formatted time code with local time zone offset to UTC information included when this configuration file was created.

number_of_processes: (required) *NX_UINT* {units=*NX_UNITLESS*}

How many individual analyses should the tool execute as part of the analysis.

PROCESS: (required) *NXprocess* <=

dataset: (required) *NXapm_input_reconstruction*

filename: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

dataset_name_reconstruction: (required) *NX_CHAR* <=

dataset_name_mass_to_charge: (required) *NX_CHAR* <=

iontypes: (required) *NXapm_input_ranging*

filename: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

group_name_iontypes: (required) *NX_CHAR* <=

spatial_filter: (optional) *NXspatial_filter*

windowing_method: (required) *NX_CHAR* <=

CG_ELLIPSOID_SET: (optional) *NXcg_ellipsoid_set* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

identifier_offset: (required) *NX_INT* <=

center: (required) *NX_NUMBER* <=

half_axes_radii: (required) *NX_NUMBER* <=

orientation: (required) *NX_NUMBER* <=

CG_CYLINDER_SET: (optional) *NXcg_cylinder_set* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

identifier_offset: (required) *NX_INT* <=

center: (required) *NX_NUMBER* <=

height: (required) *NX_NUMBER* <=

radii: (required) *NX_NUMBER* <=

CG_HEXAHEDRON_SET: (optional) *NXcg_hexahedron_set* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

identifier_offset: (required) *NX_INT* <=

hexahedra: (required) *NXcg_face_list_data_structure* <=

CS_FILTER_BOOLEAN_MASK: (optional) *NXcs_filter_boolean_mask* <=

number_of_objects: (required) *NX_UINT* <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

identifier: (required) *NX_UINT* <=

evaporation_id_filter: (optional) *NXsubsampling_filter*

iontype_filter: (optional) *NXmatch_filter*

method: (required) *NX_CHAR* <=

match: (required) *NX_NUMBER* <=

hit_multiplicity_filter: (optional) *NXmatch_filter*

edge_of_the_dataset: (required) *NXprocess*

The tool enables to inject a previously computed triangle soup or triangulated surface mesh representing a model (of the surface) of the edge of the dataset. This model can be used to detect and control various sources of bias in the analyses. **filename**: (required) *NX_CHAR*

Name of the HDF5 file which contains vertex coordinates and facet indices to describe the desired set of triangles which represents the edge of the dataset. **@version**: (required) *NX_CHAR*

Version identifier of the file such as a secure hash which documents the binary state of the file to add an additional layer of reproducibility from which file specifically contains these data.

dataset_name_vertices: (required) *NX_CHAR*

Absolute path to the HDF5 dataset in the respectively specified HDF5 file under filename which details the array of vertex positions.

dataset_name_facet_indices: (required) *NX_CHAR*

Absolute path to the HDF5 dataset in the respective specified HDF5 file under filename which details the array of facet indices.

ion_to_edge_distances: (optional) *NXprocess*

The tool enables to inject precomputed distance information for each point/ion which can be used for further post-processing and analysis. **filename**: (required) *NX_CHAR*

Name of an HDF5 file which contains the ion distances.

@version: (required) *NX_CHAR*

Version identifier of the file such as a secure hash which documents the binary state of the file to add an additional layer of reproducibility from which file specifically contains these data.

dataset_name: (required) *NX_CHAR*

Absolute HDF5 path to the dataset with distance values for each ion.

delocalization: (optional) *NXprocess*

Discretization of the ion point cloud on a three-dimensional grid.

input: (required) *NX_CHAR*

Delocalization in the field of atom probe microscopy is the process of discretizing a point cloud. By default the tool computes a full kernel density estimation of decomposed ions to create one discretized field for each element.

Although, this uses an efficient multithreaded algorithm, the computation is costly. Therefore, it can be advantageous for users to load an already computed delocalization. This can be achieved with the load_existent option. When using this option the user is responsible to assure that the settings which were used for computing this already existent delocalization are specified in the same manner as they were.

Any of these values: `default` | `load_existent`

isotope_whitelist: (required) *NX_UINT* (Rank: 2, Dimensions: [n_ityp_deloc_cand, n_ivec]) {units=*NX_UNITLESS*}

Matrix of isotope vectors representing iontypes. The filter specifies a matrix of isotope_vectors which is the most general approach to define if and how many times an ion is counted. Currently, paraprobe_nanochem performs a so-called atomic decomposition of all iontypes. Specifically, the tool interprets of how many elements/atoms a molecular ion is composed; and thus determines the atoms multiplicity with respect to the iontype.

Let's take the hydroxonium H3O+ molecular ion as an example: It contains hydrogen and oxygen as atoms. The multiplicity of hydrogen is three whereas that of oxygen is one. Therefore in an atomic decomposition computation of the iso-surface each H3O+ ion adds three hydrogen counts. This is a practical solution which accepts the situation that during an atom probe experiment not each bond of each ion/a group of neighboring atoms is broken but molecular ions get detected. The exact ab-initio details depend on the local field conditions and thus also the detailed spatial arrangement of the atoms and their own electronic state and that of the neighbors before and upon launch. Being able to measure the information for such sites only as molecular ions causes an inherent information loss with respect to the detailed spatial arrangement. This information loss is more relevant for local electrode atom probe than for field ion microscopy setting how precisely the atomic positions can be reconstructed. Accounting for multiplicities assures that at least the compositional information is analyzed.

gridresolutions: (required) *NX_FLOAT* {units=*NX_LENGTH*}

List of individual grid resolutions to analyse. Paraprobe discretizes on a cuboidal 3D grid with cubic cells, with an edge length of values in gridresolutions.

kernel_size: (required) *NX_UINT* {units=*NX_UNITLESS*}

Half the width of a $(2 \cdot n + 1)^3$ cubic kernel of voxel beyond which the Gaussian Ansatz function will be truncated. Intensity beyond the kernel is refactored into the kernel via a normalization procedure.

kernel_variance: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Variance of the Gaussian Ansatz kernel $\sigma_x = \sigma_y = 2 \cdot \sigma_z$.

normalization: (required) *NX_CHAR*

How should the results of the kernel-density estimation be computed into quantities. By default the tool computes the total number (intensity) of ions or elements. Alternatively the tool can compute the total intensity, the composition, or the concentration of the ions/elements specified by the white list of elements in each voxel.

Any of these values:

- `total`
- `candidates`
- `composition`
- `concentration`

has_scalar_fields: (required) *NX_BOOLEAN*

Specifies if the tool should report the delocalization 3D field values.

isosurfacing: (optional) *NXprocess*

Optional computation of iso-surfaces after each computed delocalization to identify for instance objects in the microstructure (line features, interfaces, precipitates). **edge_handling_method**: (required) *NX_CHAR*

As it is detailed in M. Kühbach et al. 2022 npj Comp. Mat., the handling of triangles at the edge of the dataset requires special attention. Especially for composition-normalized delocalization it is possible that the composition increases towards the edge of the dataset because the quotient of two numbers which are both smaller than one is larger instead of smaller than the counter. By default, the tool uses a modified marching cubes algorithm of Lewiner et al. which detects if voxels face such a situation. In this case, no triangles are generated for such voxels. Alternatively, (via setting keep_edge_triangles) the user can instruct the tool to not remove these triangles at the cost of bias.

Specifically, in this case the user should understand that all objects/microstructural features in contact with the edge of the dataset get usually artificial enlarged and their surface mesh often closed during the marching. This closure however is artificial! It can result in biased shape analyses for those objects. The reason why this should in general be avoided is a similar argument as when one analyzes grain shapes in orientation microscopy via e.g. SEM/EBSD. Namely, these grains, here the objects at the edge of the dataset, were not fully captured during e.g. limited field of view. Therefore, it is questionable if one would like to make substantiated quantitative statements about them.

Thanks to collaboration with the V. V. Rielli and S. Primig, though, paraprobe-nanochem implements a complete pipeline to process even these objects at the edge of the dataset. Specifically, the objects are replaced by so-called proxies, i.e. replacement objects whose holes on the surface mesh have been closed if possible via iterative mesh and hole-filling procedures with fairing operations. In the results of each paraprobe-nanochem run, these proxy objects are listed separately to allow users to quantify and analyze in detail the differences when accounting for these objects or not. Especially this is relevant in atom probe microscopy as objects can contain a few dozen atoms only. Users should be aware that results from fairing operations should be compared to results from analyses where all objects at the edge of the dataset have been removed.

Also users should be careful with overestimating the statistical significance of their dataset especially when using atom probe to compare multiple descriptors: Even though a dataset may give statistically significant results for compositions, this does not necessarily mean it will yield also statistically significant and unbiased results for three-dimensional object analyses. Being able to quantify these effects and making atom probers aware of these subtleties was one of the main reasons why the paraprobe-nanochem tool was implemented.

Any of these values: `default` | `keep_edge_triangles`

edge_threshold: (required) *NX_FLOAT* {units=*NX_LENGTH*}

The ion-to-edge-distance that is used in the analyses of objects (and proxies) to identify whether these are inside the dataset or close to the edge of the dataset. If an object has at least one ion with an ion-to-edge-distance below this threshold, the object is considered as one which lies close to the edge of the dataset. This implements essentially a distance-based approach to solve the in general complicated and involved treatment of computing volumetric intersections between not-necessarily convex closed 2-manifolds. In fact, such computational geometry analyses can face numerical robustness issues as a consequence of which a mesh can be detected as lying completely inside a dataset although in reality it is epsilon-close only, i.e. almost touching only the edge (e.g. from inside). Practically, humans would state in such case that the object is close to the edge of the dataset; however mathematically the object is indeed completely inside. In short, a distance-based approach is rigorous and more flexible.

phi: (required) *NX_FLOAT* {units=*NX_ANY*}

Array of iso-contour values. For each value the tool computes an iso-surface and performs subsequent analyses. The unit depends on the choice for the normalization of the accumulated ion intensity values per voxel:

- total, total number of ions, irrespective their iontype
- candidates, total number of ions with type in the iso-

tope_whitelist.

- composition, candidates but normalized by composition, i.e. at.-%
- concentration, candidates but normalized by voxel volume, i.e. ions/nm^3

has_triangle_soup: (required) *NX_BOOLEAN*

Specifies if the tool should report the triangle soup which represents each triangle of the iso-surface complex. Each triangle is reported with an ID specifying to which triangle cluster (with IDs starting at zero) the triangle belongs. The clustering is performed with a modified DBScan algorithm.

has_object: (required) *NX_BOOLEAN*

Specifies if the tool should analyze for each cluster of triangles how they can be combinatorially processed to describe a closed polyhedron. Such a closed polyhedron (not-necessarily convex!) can be used to describe objects with relevance in the microstructure. Users should be aware that the resulting mesh does not necessarily represent the original precipitate. In fact, inaccuracies in the reconstructed positions cause inaccuracies in all downstream processing operations. Especially the effect on one-dimensional spatial statistics like nearest neighbor correlation functions these effects were discussed in the literature B. Gault et al. In continuation of these thoughts this applies also to reconstructed objects. A well-known example is the discussion of shape deviations of Al3Sc precipitates in aluminium alloys which in reconstructions can appear as ellipsoids although they should be almost spherical, depending on their size.

has_object_geometry: (required) *NX_BOOLEAN*

Specifies if the tool should report a triangulated surface mesh for each identified closed polyhedron. It is common that a marching cubes algorithm creates iso-surfaces with a fraction of very small sub-complexes (e.g. small isolated tetrahedra).

These can be for instance be small tetrahedra/polyhedra about the center of a voxel of the support grid on which marching cubes operates. When these objects are small, it is possible that they contain no ion; especially when considering that delocalization procedures smoothen the positions of the ions. Although these small objects are interesting from a numerical point of view, scientists may argue they are not worth to be reported: Physically a microstructural feature should contain at least a few atoms to become relevant. Therefore, paraprobe-nanochem by default does not report closed objects which bound not at least one ion.

has_object_properties: (required) *NX_BOOLEAN*

Specifies if the tool should report properties of each closed polyhedron, such as volume and other details.

has_object_obb: (required) *NX_BOOLEAN*

Specifies if the tool should report for each closed polyhedron an approximately optimal bounding box fitted to all triangles of the surface mesh of the object and ion positions inside or on the surface of the mesh. This bounding box informs about the closed object's shape (aspect ratios).

has_object_ions: (required) *NX_BOOLEAN*

Specifies if the tool should report for each closed polyhedron all evaporation IDs of those ions which lie inside or on the boundary of the polyhedron. This information can be used e.g. in the paraprobe-intersector tool to infer if two objects share common ions, which can be interpreted as an argument to assume that the two objects intersect.

Users should be aware that two arbitrarily closed polyhedra in three-dimensional space can intersect but not share a common ion. In fact, the volume bounded by the polyhedron has sharp edges. When taking two objects, an edge of one object may for instance pierce into the surface of another object. In this case the objects partially overlap / intersect volumetrically; however this piercing might be so small or happening in the volume between two ion positions and thus sharing ions is a sufficient but not a necessary condition for object intersections.

Paraprobe-intersector implements a rigorous alternative to handle such intersections using a tetrahedralization of closed objects. However, in many practical cases, we found through examples that there are polyhedra (especially when they are non-convex and have almost point-like) connected channels, where tetrahedralization libraries have challenges dealing with. In this case checking intersections via shared_ions is a more practical alternative.

has_object_edge_contact: (required) *NX_BOOLEAN*

Specifies if the tool should report if a (closed) object has contact with the edge of the dataset. For this the tool currently inspects if the shortest distance between the set of triangles of the surface mesh and the triangles of the edge model is larger than the edge_threshold. If this is the case, the object is assumed to be deeply embedded in the interior of the dataset. Otherwise, the object is considered to have an edge contact, i.e. it is likely affected by the fact that the dataset is finite.

has_proxy: (required) *NX_BOOLEAN*

Specifies if the tool should analyze a doppelganger/proxy mesh for each cluster of triangles whose combinatorial analysis according to has_object showed that the object is not a closed polyhedron. Such proxies are closed via iterative hole-filling, mesh refinement, and fairing operations. Users should be aware that the resulting mesh does not necessarily represent the original precipitate. In most cases objects, like precipitates in atom probe end up as open objects because they have been clipped by the edge of the dataset. Using a proxy is then a strategy to still be able to account for these objects. Nevertheless users should make themselves familiar with the potential consequences and biases which this can introduce into the analysis.

has_proxy_geometry: (required) *NX_BOOLEAN*

Like has_object_geometry but for the proxies.

has_proxy_properties: (required) *NX_BOOLEAN*

Like has_object_properties but for the proxies.

has_proxy_obb: (required) *NX_BOOLEAN*

Like has_object_obb but for the proxies.

has_proxy_ions: (required) *NX_BOOLEAN*

Like has_object_ions but for the proxies.

has_proxy_edge_contact: (required) *NX_BOOLEAN*

Like has_object_edge_contact but for the proxies.

has_object_auto_proxigram: (required) *NX_BOOLEAN*

Specifies if the tool should report for each closed object a (cylindrical) region of interest placed, centered, and align with the local normal for each triangle of the object.

has_object_auto_proxigram_edge_contact: (required) *NX_BOOLEAN*

Specifies if the tool should report for each ROI that was placed at a triangle of each object if this ROI intersects the edge of the dataset. Currently paraprobe-nanochem supports cylindrical ROIs. A possible intersection of these with the edge of the dataset, i.e. the triangulated surface mesh model for the edge is performed. This test checks if the cylinder intersects with a triangle of the surface mesh. If this is the case, the ROI is assumed to make edge contact, else, the ROI is assumed to have no edge contact.

This approach does not work if the ROI would be completely outside the dataset. Also in this case there would be no intersection. For atom probe this case is practically irrelevant because for such a ROI there would also be no ion laying inside the ROI. Clearly it has thus to be assumed that the edge model culls the entire dataset. Instead, if one would cut a portion of the dataset, compute an edge model for this point cloud, it might make sense to place a ROI but in this case the edge contact detection is not expected to work properly.

interfacial_excess: (optional) *NXprocess*

Analyses of interfacial excess.

interface_model: (required) *NX_CHAR*

Interfacial excess computations are performed for local regions-of-interests (ROIs) at selected facets or interface patch. For instance many scientist compute the interfacial excess for selected triangle facets of a created iso-surface. In this case, computed iso-surfaces of paraprobe could be used. An example are triangle facet sets about closed polyhedra, for instance to compute interfacial excess related to phase boundaries of second-phase precipitates.

Another example are free-standing triangle patches of the iso- surfaces which paraprobe creates. These could be characterized for interfacial excess. The sub-routines during iso-surface computations already include a procedure to automatically align local triangle normals based on the gradients of e.g. composition fields. In this case, these triangulated surface patches could also be used as a source for computing interfacial excess.

Often scientists face situations, though, in which there is no immediately evident composition gradient across the interface (grain or phase boundary) and orientation information about the adjoining crystal is neither available nor reliable enough.

In this case P. Felfer et al. proposed a method to manually place control points and run an automated tessellation-based algorithm to create a triangulated surface patch, i.e. a model of the location of the interface. In a post-processing step this triangle set can then be used to compute again interfacial excess in an automated manner by placing ROIs and aligning them with consistently precomputed triangle normals.

A similar use case is conceptually the one proposed by X. Zhou et al. They used first a deep-learning method to locate planar triangulated grain boundary patches. These are eventually processed further with manual editing of the mesh via tools like Blender. Once the user is satisfied with the mesh, the computations of interfacial excess reduce again to an automated placing of ROIs, computations of the distributing of ions to respective ROIs and reporting the findings via plotting.

Yet another approach for constructing an triangulated surface patch of an interface is to use point cloud processing methods which have been proposed in the laser-scanning, geoinformatics, and CAD community. Different computational geometry methods are available for fitting a parameterized surface to a set of points, using e.g. non-uniform rational B-splines (NURBS) and triangulating these according to prescribed mesh quality demands.

The advantage of these methods is that they can be automated and pick up curved interface segments. The disadvantage is their often strong sensitivity to parameterization. As a result also such methods can be post-processed to yield a triangulated surface patch, and thus enable to run again automated ROI placement methods. For example like these which were explored for the use case of iso-surfaces with closed objects and free-standing surface patches that delineate regions of the dataset with a pronounced composition gradient normal to the interface.

This summary of the situations which atom probers can face when requesting for interfacial excess computations, substantiates there exists a common set of settings which can describe all of these methods and, specifically, as here exemplified, the automated placing and alignment functionalities for ROIs that is an important step all these workflows.

Specifically, paraprobe-nanochem operates on an already existent triangle set.

Any of these values: `isosurface` | `external`

external: (optional) *NXprocess*

The interface model is the result of a previous (set of) processing steps as a result of which the user has created a triangulated surface mesh (or a set of, eventually connected such meshes). These interface models are useful, if not required, in cases when there is no other independent approach to locate an interface.

These are cases when insufficient crystallographic latent information is available and also no consistent concentration gradient detectable across the interface. It is then the users' responsibility to deliver a triangle mesh of the interface model. **file_name**: (required) *NX_CHAR*

Filename to HDF5 file which contain vertex coordinates, facet indices, facet unit normals. The user is responsible for the triangle and winding order to be consistent. Input is expected as a matrix of the coordinates for all disjoint vertices, a (Nvertices, 3)-shaped array of NX_FLOAT. Input is expected to include also a matrix of facet indices referring to these disjoint vertices. This matrix should be a (Nfacets, 3)-shaped array of NX_UINT. Further required input is a (Nfacets, 3)-shaped array of NX_FLOAT signed facet unit normals and a (Nvertices, 3)-shaped array of NX_FLOAT signed vertex unit normals. Vertex indices need to start at zero and must not exceed Nvertices - 1, i.e. the identifier_offset is 0 and facet indices are indexed implicitly, i.e. [0, Nvertices-1]. **@version**: (required) *NX_CHAR*

Version identifier of the file such as a secure hash which documents the binary state of the file to add an additional layer of reproducibility from which file specifically contains these data.

dataset_name_vertices: (required) *NX_CHAR*

Absolute HDF5 path to the dataset which specifies the array of vertex positions.

dataset_name_facet_indices: (required) *NX_CHAR*

Absolute HDF5 path to the dataset which specifies the array of facet indices.

dataset_name_facet_normals: (required) *NX_CHAR*

Absolute HDF5 path to the dataset which specifies the array of facet signed unit normals.

dataset_name_facet_vertices: (required) *NX_CHAR*

Absolute HDF5 path to the dataset which specifies the array of vertex signed unit normals.

Users should be aware that triangulated surface meshes are only approximations to a given complex, eventually curved shape. Consequently, computations of normals show differences between the vertex and facet normals. Vertex normals have to be interpolated from normals of neighboring facets.

Consequently, these normals are affected by the underlying parameterization and curvature estimation algorithms, irrespective of how contributions from neighboring facets are weighted. By contrast, facet normals are clearly defined by the associated triangle. Their disadvantage is that they the normal field has discontinuities at the edges. In general the coarser an object is triangulated the more significant the difference becomes between computations based on facet or vertex normals. Paraprobe-nanochem works with facet normals as it can use parts of the numerical performance gained by using cutting edge libraries to work rather with finer meshes.

interface_meshing: (optional) *NXprocess*

Create a simple principle component analysis (PCA) to mesh a free-standing interface patch through a point cloud of decorating solutes. These models can be useful for quantification of Gibbsian interfacial excess for interfaces where iso-surface based methods may fail or closed objects from iso-surfaces are not desired or when e.g. there are no substantial or consistently oriented concentration gradients across the interface patch.

The interface_meshing functionality of paraprobe-nanochem can be useful when there is also insufficient latent crystallographic information available that could otherwise support modelling the interface, via e.g. ion density traces in field-desorption maps, as were used and discussed by Y. Wei et al. or are discussed by A. Breen et al.

It is noteworthy that the method here used is conceptually very similar in implementation to the work by Z. Peng et al. Noteworthy, her team uses the DCOM approach originally proposed by P. Felfer et al. However, both of these previous works neither discuss in detail nor implement inspection functionalities which enable a detection of potential geometric inconsistencies or self-interactions of the resulting DCOM mesh. This is what paraprobe-nanochem implements via the Computational Geometry Algorithms Library.

initialization: (required) *NX_CHAR*

Method how to initialize the PCA:

- default, means based on segregated solutes in the ROI
- control_point_file, means based on reading an external list of control points, currently coming from the Leoben APT_Analyzer.

The control_point_file is currently expected with a specific format. The Leoben group lead by L. Romaner has developed a GUI tool A. Reichmann et al. to create a control_point_file which can be parsed by paraprobe-parmsetup to match the here required formatting in control_points.

Any of these values: `default` | `control_point_file`

filename: (required) *NX_CHAR*

The name of the control point file to use.

@version: (required) *NX_CHAR*

Version identifier of the file such as a secure hash which documents the binary state of the file to add an additional layer of reproducibility from which file specifically contains these data.

control_points: (required) *NX_FLOAT* (Rank: 2, Dimensions: [N, n_control_pts]) {units=*NX_LENGTH*}

X, Y, Z coordinates of disjoint control point read from an HDF5 file named according to control_point_file.

method: (required) *NX_CHAR*

Method used for identifying and refining the location of the interface. Currently, paraprobe-nanochem implements a PCA followed by an iterative loop of isotropic mesh refinement and DCOM step(s), paired with self-intersection detection in a more robust implementation.

Obligatory value: `pca_plus_dcom`

number_of_iterations: (required) *NX_UINT* {units=*NX_UNITLESS*}

How many times should the DCOM and mesh refinement be applied?

target_edge_length: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_fct_iterations]) {units=*NX_LENGTH*}

Array of decreasing positive not smaller than one nanometer real values which specify how the initial triangles of the mesh should be iteratively refined by edge splitting and related mesh refinement operations.

target_dcom_radius: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_fct_iterations]) {units=*NX_LENGTH*}

Array of decreasing positive not smaller than one nanometer real values which specify the radius of the spherical region of interest within which the DCOM algorithm decides for each vertex how the vertex will be eventually relocated. The larger the DCOM radius is relative to the target_edge_length the more likely it is that vertices will be relocated so substantially that eventually triangle self-intersections can occur. If the code detects these it warns and stops in a controlled manner so that the user can repeat the analyses with a smaller value.

target_smoothing_step: (required) *NX_UINT* (Rank: 1, Dimensions: [n_fct_iterations]) {units=*NX_UNITLESS*}

Array of integers which specify for each DCOM step how many times the mesh should be iteratively smoothened.

Users should be aware the three array target_edge_length, target_dcom_radius, and target_smoothing_step are interpreted in the same sequence, i.e. the zeroth entry of each array specifies the values to be used in the first DCOM iteration. The first entry of each array those for the second DCOM iteration and so on and so forth.

decorating_iontypes_filter: (required) *NXprocess*

Specify the types of those ions which decorate the interface and can thus be assumed as markers for locating the interface and refining its local curvature. **candidates**: (required) *NX_UINT* (Rank: 1, Dimensions: [n_fct_filter_cand]) {units=*NX_UNITLESS*}

Array of iontypes to filter. The list is interpreted as a whitelist, i.e. ions of these types are considered the decorating species (solutes).

composition_profiling: (optional) *NXprocess*

Functionalities for placing regions-of-interest (ROIs) in the dataset or at specific microstructural features to characterize composition profiles and cumulated profiles for quantification of interfacial excess. Paraprobe-nanochem currently places cylindrical ROIs. ROIs are probed across the triangulated surface of a user-defined mesh. ROIs are placed at the barycenter of the triangular facet.

The tool can be instructed to orient the profile for each ROIs with the positive normal of the triangle facet normals. Profiles are computed for each ROI and facet triangle. The code will test which ROIs are completely embedded in the dataset. Specifically, in this test the tool evaluates if the ROI cuts at least one triangle of the triangulated surface mesh of the edge of the dataset. If this is the case the ROI will be considered close to the edge (of the dataset) and not analyzed further; else the ROI will be processed further. Users should be aware that the latter intersection analysis is strictly speaking not a volumetric intersection analysis as such one is much more involved because the edge model can be a closed non-convex polyhedron in which case one would have to test robustly if the cylinder pierces or is laying completely inside the polyhedron. For this the polyhedron has to be tessellated into convex polyhedra as otherwise tests like the Gilbert-Johnson-Keerthi algorithm would not be applicable.

Specifically, the tool computes atomically decomposed profiles. This means molecular ions are split into atoms/isotopes with respective multiplicity. As an example an H3O+ molecular ion contains three hydrogen and one oxygen atom respectively. The tool then evaluates how many ions are located inside the ROI or on the surface of the ROI respectively. All atom types and the unranged ions are distinguished. As a result, the analyses yield for each ROI a set of sorted lists of signed distance values. Currently, the distance is the projected distance of the ion position to the barycenter of the triangle and triangle plane.

This will return a one-dimensional profile. Post-processing the set of atom-type-specific profiles into cumulated profiles enable the classical Krakauer/Seidman-style interfacial excess analyses. Furthermore, the tool can be instructed to compute for each (or a selected sub-set of facet) a set of differently oriented profiles. **distancing_model**: (required) *NX_CHAR*

Which type of distance should be reported for the profile.

Obligatory value: `project_to_triangle_plane`

direction_model: (required) *NX_CHAR*

In which directions should the tool probe for each ROI.

Obligatory value: `triangle_outer_unit_normal`

roi_cylinder_height: (required) *NX_FLOAT* {units=*NX_LENGTH*}

For each ROI, how high (projected on the cylinder axis) should the cylindrical ROI be.

roi_cylinder_radius: (required) *NX_FLOAT* {units=*NX_LENGTH*}

For each ROI, how wide (radius) should the cylindrical ROI be.

feature_mesh: (required) *NXprocess*

The feature mesh enables the injection of previously computed triangle soup or mesh data. Such a mesh can be the model for a grain- or phase boundary patch (from e.g. interface_meshing) jobs. **filename**: (required) *NX_CHAR*

Name of the HDF5 file which contains vertex coordinates and facet indices to describe the desired set of triangles which represents the feature. **@version**: (required) *NX_CHAR*

Version identifier of the file such as a secure hash which documents the binary state of the file to add an additional layer of reproducibility from which file specifically contains these data.

dataset_name_vertices: (required) *NX_CHAR*

Absolute path to the HDF5 dataset in the respectively specified HDF5 file under filename which details the array of vertex positions.

dataset_name_facet_indices: (required) *NX_CHAR*

Absolute path to the HDF5 dataset in the respective specified HDF5 file under filename which details the array of facet indices.

dataset_name_facet_normals: (required) *NX_CHAR*

Absolute path to the HDF5 dataset in the respective specified HDF5 file under filename which details consistently oriented facet normals of the facets.

patch_identifier_filter: (optional) *NXmatch_filter*

method: (required) *NX_CHAR <=*

match: (required) *NX_NUMBER <=*

ion_to_feature_distances: (optional) *NXprocess*

The tool enables to inject precomputed distance information for each point which can be used for further post-processing and analysis. **filename**: (required) *NX_CHAR*

Name of an HDF5 file which contains ion distances.

@version: (required) *NX_CHAR*

Version identifier of the file such as a secure hash which documents the binary state of the file to add an additional layer of reproducibility from which file specifically contains these data.

dataset_name: (required) *NX_CHAR*

Absolute HDF5 path to the dataset with distance values for each ion.

performance: (required) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR <=*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_config_nanochem/ENTRY-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/analysis_description-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/definition-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/number_of_processes-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/performance-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/direction_model-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/distancing_model-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/feature_mesh-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/feature_mesh/dataset_name_facet_indices-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/feature_mesh/dataset_name_facet_normals-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/feature_mesh/dataset_name_vertices-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/feature_mesh/filename-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/feature_mesh/filename@version-attribute*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/feature_mesh/patch_identifier_filter-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/feature_mesh/patch_identifier_filter/match-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/feature_mesh/patch_identifier_filter/method-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/ion_to_feature_distances-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/ion_to_feature_distances/dataset_name-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/ion_to_feature_distances/filename-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/ion_to_feature_distances/filename@version-attribute*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/roi_cylinder_height-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/composition_profiling/roi_cylinder_radius-field*

- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/dataset-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/dataset/dataset_name_mass_to_charge-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/dataset/dataset_name_reconstruction-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/dataset/filename-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/dataset/filename@version-attribute*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/gridresolutions-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/has_scalar_fields-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/input-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/interfacial_excess-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/interfacial_excess/external-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/interfacial_excess/external/dataset_name_facet_indices-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/interfacial_excess/external/dataset_name_facet_normal field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/interfacial_excess/external/dataset_name_facet_vertices field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/interfacial_excess/external/dataset_name_vertices-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/interfacial_excess/external/file_name-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/interfacial_excess/external/file_name@version-attribute*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/interfacial_excess/interface_model-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/edge_handling_method-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/edge_threshold-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/has_object-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/has_object_auto_proxigram-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/has_object_auto_proxigram_edge_contact field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/has_object_edge_contact-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/has_object_geometry-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/has_object_ions-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/has_object_obb-field*

- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/ion_to_edge_distances-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/ion_to_edge_distances/dataset_name-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/ion_to_edge_distances/filename-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/ion_to_edge_distances/filename@version-attribute*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/iontype_filter-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/iontype_filter/match-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/iontype_filter/method-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/iontypes-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/iontypes/filename-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/iontypes/filename@version-attribute*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/iontypes/group_name_iontypes-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/cardinality-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/center-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/dimensionality-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/height-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/identifier_offset-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/radii-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/cardinality-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/center-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/dimensionality-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/half_axes_radii-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/identifier_offset-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/orientation-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/cardinality-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/dimensionality-field*

- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/hexahedra-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/identifier_offset-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK-group*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/bitdepth-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/identifier-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/mask-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/number_of_objects-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/spatial_filter/windowing_method-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/program-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/program@version-attribute*
- */NXapm_paraprobe_config_nanochem/ENTRY/results_path-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/time_stamp-field*
- */NXapm_paraprobe_config_nanochem/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_config_nanochem.nxdl.xml

NXapm_paraprobe_config_ranger

Status:

application definition, extends *NXobject*

Description:

Configuration of a paraprobe-ranger tool run in atom probe microscopy.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_isotopes: The number of isotopes to consider as building blocks for searching molecular ions.

n_composition: The number of compositions to consider for molecular ion search tasks.

Groups cited:
NXapm_input_ranging, *NXapm_input_reconstruction*, *NXcg_cylinder_set*, *NXcg_ellipsoid_set*, *NXcg_face_list_data_structure*, *NXcg_hexahedron_set*, *NXcs_filter_boolean_mask*, *NXcs_profiling*, *NXentry*, *NXmatch_filter*, *NXprocess*, *NXspatial_filter*, *NXsubsampling_filter*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_config_ranger`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

time_stamp: (required) *NX_DATE_TIME*

ISO 8601 formatted time code with local time zone offset to UTC information included when this configuration file was created.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

number_of_processes: (required) *NX_UINT* {units=*NX_UNITLESS*}

How many task to perform?

PROCESS: (optional) *NXprocess* <=

apply_existent_ranging: (optional) *NXprocess*

dataset: (required) *NXapm_input_reconstruction*

filename: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

dataset_name_reconstruction: (required) *NX_CHAR* <=

dataset_name_mass_to_charge: (required) *NX_CHAR* <=

iontypes: (required) *NXapm_input_ranging*

filename: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

group_name_iontypes: (required) *NX_CHAR* <=

spatial_filter: (optional) *NXspatial_filter*

windowing_method: (required) *NX_CHAR* <=

CG_ELLIPSOID_SET: (optional) *NXcg_ellipsoid_set* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

identifier_offset: (required) *NX_INT* <=

center: (required) *NX_NUMBER* <=

half_axes_radii: (required) *NX_NUMBER* <=

orientation: (required) *NX_NUMBER* <=

CG_CYLINDER_SET: (optional) *NXcg_cylinder_set* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

identifier_offset: (required) *NX_INT* <=

center: (required) *NX_NUMBER* <=

height: (required) *NX_NUMBER* <=

radii: (required) *NX_NUMBER* <=

CG_HEXAHEDRON_SET: (optional) *NXcg_hexahedron_set* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

identifier_offset: (required) *NX_INT* <=

hexahedra: (required) *NXcg_face_list_data_structure* <=

CS_FILTER_BOOLEAN_MASK: (optional) *NXcs_filter_boolean_mask* <=

number_of_objects: (required) *NX_UINT* <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

identifier: (required) *NX_UINT* <=

evaporation_id_filter: (optional) *NXsubsampling_filter*

iontype_filter: (optional) *NXmatch_filter*

hit_multiplicity_filter: (optional) *NXmatch_filter*

molecular_ion_search: (optional) *NXprocess*

assumed_composition_isotopes: (optional) *NX_UINT* (Rank: 2, Dimensions: [n_composition, 2]) {units=*NX_UNITLESS*}

A list of pairs of number of protons and either the value 0 (per row) or the mass number for all those isotopes which are assumed present in a virtual specimen. The purpose of this field is to compute also composition-weighted products to yield a simple estimation which could potentially help scientists to judge if certain molecular ions are to be expected. The corresponding setting store_composition_weighted_product should be activated.

isotope_whitelist: (required) *NX_UINT* (Rank: 2, Dimensions: [n_isotopes, 2]) {units=*NX_UNITLESS*}

A list of pairs of number of protons and mass number for all isotopes to consider that can be composed into (molecular) ions, during the recursive molecular_ion_search.

mass_to_charge_interval: (required) *NX_FLOAT* (Rank: 1, Dimensions: [2]) {units=*NX_ANY*}

The mass-to-charge-state ratio interval in which all molecular ions are searched.

maximum_charge: (required) *NX_UINT* {units=*NX_UNITLESS*}

The maximum charge that a molecular ion should have.

maximum_number_of_isotopes: (required) *NX_UINT* {units=*NX_UNITLESS*}

The maximum number of isotopes of which the molecular ion should be composed. Currently this must not be larger than 32.

Users should be warned that the larger the maximum_charge and especially the larger the maximum_number_of_isotopes is chosen, the eventually orders of magnitude more costly the search becomes.

This is because paraprobe-ranger computes really all (at least) theoretically possible combinations that would have likely a mass-to-charge-state ratio in the specified mass_to_charge_interval. It is the challenge in atom probe to judge which of these (molecular) ions are feasible and also practically possible. This tool does not answer this question.

Namely, which specific molecular ion will evaporate, remain stable during flight and becomes detected is a complicated and in many cases not yet in detail understood phenomenon. The ab-initio conditions before and during launch, the local environment, arrangement and field as well as the flight phase in an evacuated but not analysis chamber with a complex electrical field, eventual laser pulsing in place, temperature and remaining atoms or molecules all can have an effect which iontypes are really physically evaporating and detected.

store_atomic_mass_sum: (required) *NX_BOOLEAN*

Report the accumulated atomic mass from each isotope building the ion. Accounts for each identified ion. Relatistic effects are not accounted for.

store_natural_abundance_product: (required) *NX_BOOLEAN*

Report the product of the natural abundances from each isotope building the ion. Accounts for each identified ion.

The value zero indicates it is not possible to build such molecular ion from nuclids which are all observationally stable. Very small values can give an idea/about how likely such a molecular ion is expected to form assuming equal probabilities.

However in atom probe experiments this product has to be modified by the (spatially-correlated) local composition in the region from which the ions launch because the formation of a molecular ion depends as summarized under maximum_number_of_isotopes on the specific quantum-mechanical configuration and field state upon launch or/and (early state) of flight respectively. We are aware that this modified product can have a substantially different value than the natural_abundance_product.

Natural abundancies folded with the estimated compositions of the specimen can differ by orders of magnitude.

store_charge_state: (required) *NX_BOOLEAN*

Report the charge state of the ions.

store_disjoint_isotopes: (required) *NX_BOOLEAN*

Report if identified ions should be characterized wrt to their number of disjoint isotopes.

check_existent_ranging: (optional) *NXprocess*

iontypes: (required) *NXapm_input_ranging*

filename: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

group_name_iontypes: (required) *NX_CHAR* <=

performance: (required) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_config_ranger/ENTRY-group*
- */NXapm_paraprobe_config_ranger/ENTRY/analysis_description-field*
- */NXapm_paraprobe_config_ranger/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_config_ranger/ENTRY/definition-field*
- */NXapm_paraprobe_config_ranger/ENTRY/number_of_processes-field*
- */NXapm_paraprobe_config_ranger/ENTRY/performance-group*
- */NXapm_paraprobe_config_ranger/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS-group*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging-group*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/dataset-group*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/dataset/dataset_name_mass_to_charge-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/dataset/dataset_name_reconstruction-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/dataset/filename-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/dataset/filename@version-attribute*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/evaporation_id_filter-group*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/hit_multiplicity_filter-group*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/iontype_filter-group*

- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/iontypes-group*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/iontypes/filename-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/iontypes/filename@version-attribute*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/iontypes/group_name_iontypes-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter-group*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_CYLINDER_SET-group*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_CYLINDER_SET/cardinality-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_CYLINDER_SET/center-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_CYLINDER_SET/dimensionalit field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_CYLINDER_SET/height-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_CYLINDER_SET/identifier_offs field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_CYLINDER_SET/radii-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_ELLIPSOID_SET-group*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_ELLIPSOID_SET/cardinality-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_ELLIPSOID_SET/center-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_ELLIPSOID_SET/dimensionalit field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_ELLIPSOID_SET/half_axes_rad field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_ELLIPSOID_SET/identifier_offs field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_ELLIPSOID_SET/orientation-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_HEXAHEDRON_SET-group*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_HEXAHEDRON_SET/cardinali field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_HEXAHEDRON_SET/dimensio field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_HEXAHEDRON_SET/hexahedr group*

- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_HEXAHEDRON_SET/identifier field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CS_FILTER_BOOLEAN_MASK-group*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CS_FILTER_BOOLEAN_MASK/bitd field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CS_FILTER_BOOLEAN_MASK/ider field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CS_FILTER_BOOLEAN_MASK/mas field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CS_FILTER_BOOLEAN_MASK/num field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/windowing_method-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/check_existent_ranging-group*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/check_existent_ranging/iontypes-group*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/check_existent_ranging/iontypes/filename-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/check_existent_ranging/iontypes/filename@version-attribute*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/check_existent_ranging/iontypes/group_name_iontypes-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/molecular_ion_search-group*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/molecular_ion_search/assumed_composition_isotopes-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/molecular_ion_search/isotope_whitelist-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/molecular_ion_search/mass_to_charge_interval-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/molecular_ion_search/maximum_charge-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/molecular_ion_search/maximum_number_of_isotopes-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/molecular_ion_search/store_atomic_mass_sum-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/molecular_ion_search/store_charge_state-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/molecular_ion_search/store_disjoint_isotopes-field*
- */NXapm_paraprobe_config_ranger/ENTRY/PROCESS/molecular_ion_search/store_natural_abundance_product-field*
- */NXapm_paraprobe_config_ranger/ENTRY/program-field*
- */NXapm_paraprobe_config_ranger/ENTRY/program@version-attribute*
- */NXapm_paraprobe_config_ranger/ENTRY/results_path-field*
- */NXapm_paraprobe_config_ranger/ENTRY/time_stamp-field*
- */NXapm_paraprobe_config_ranger/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_config_ranger.nxdl.xml

NXapm_paraprobe_config_selector

Status:

application definition, extends *NXobject*

Description:

Configuration of a paraprobe-selector tool run in atom probe microscopy.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXapm_input_ranging, *NXapm_input_reconstruction*, *NXcg_cylinder_set*, *NXcg_ellipsoid_set*, *NXcg_face_list_data_structure*, *NXcg_hexahedron_set*, *NXentry*, *NXmatch_filter*, *NXprocess*, *NXspatial_filter*, *NXsubsampling_filter*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR* *<=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_config_selector`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

time_stamp: (required) *NX_DATE_TIME*

ISO 8601 formatted time code with local time zone offset to UTC information included when this configuration file was created.

number_of_processes: (required) *NX_UINT* {units=*NX_UNITLESS*}

How many roi_selection processes should the tool execute.

roi_selection: (required) *NXprocess* *<=*

This process identifies which of the points/ions in the datasets are inside or on the surface of geometric primitives and meet optionally specific other filtering constraints. A typical use case of a roi_selection is to restrict analyses to specific regions of the dataset, eventually regions with a complicated shape. **dataset**: (required) *NXapm_input_reconstruction*

filename: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

dataset_name_reconstruction: (required) *NX_CHAR* <=

dataset_name_mass_to_charge: (required) *NX_CHAR* <=

iontypes: (required) *NXapm_input_ranging*

filename: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

group_name_iontypes: (required) *NX_CHAR* <=

spatial_filter: (required) *NXspatial_filter*

windowing_method: (required) *NX_CHAR* <=

CG_ELLIPSOID_SET: (optional) *NXcg_ellipsoid_set* <=

cardinality: (required) *NX_POSINT* <=

center: (required) *NX_NUMBER* <=

half_axes_radii: (required) *NX_NUMBER* <=

CG_CYLINDER_SET: (optional) *NXcg_cylinder_set* <=

cardinality: (required) *NX_POSINT* <=

center: (required) *NX_NUMBER* <=

height: (required) *NX_NUMBER* <=

radii: (required) *NX_NUMBER* <=

CG_HEXAHEDRON_SET: (optional) *NXcg_hexahedron_set* <=

cardinality: (required) *NX_POSINT* <=

hexahedra: (required) *NXcg_face_list_data_structure* <=

vertices: (required) *NX_FLOAT*

evaporation_id_filter: (optional) *NXsubsampling_filter*

linear_range_min_incr_max: (required) *NX_UINT* <=

iontype_filter: (optional) *NXmatch_filter*

method: (required) *NX_CHAR* <=

match: (required) *NX_NUMBER* <=

hit_multiplicity_filter: (optional) *NXmatch_filter*

method: (required) *NX_CHAR* <=

match: (required) *NX_NUMBER* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_config_selector/ENTRY-group*
- */NXapm_paraprobe_config_selector/ENTRY/analysis_description-field*
- */NXapm_paraprobe_config_selector/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_config_selector/ENTRY/definition-field*
- */NXapm_paraprobe_config_selector/ENTRY/number_of_processes-field*
- */NXapm_paraprobe_config_selector/ENTRY/program-field*
- */NXapm_paraprobe_config_selector/ENTRY/program@version-attribute*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection-group*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/dataset-group*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/dataset/dataset_name_mass_to_charge-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/dataset/dataset_name_reconstruction-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/dataset/filename-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/dataset/filename@version-attribute*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/evaporation_id_filter-group*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/evaporation_id_filter/linear_range_min_incr_max-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/hit_multiplicity_filter-group*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/hit_multiplicity_filter/match-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/hit_multiplicity_filter/method-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/iontype_filter-group*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/iontype_filter/match-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/iontype_filter/method-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/iontypes-group*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/iontypes/filename-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/iontypes/filename@version-attribute*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/iontypes/group_name_iontypes-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter-group*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter/CG_CYLINDER_SET-group*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter/CG_CYLINDER_SET/cardinality-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter/CG_CYLINDER_SET/center-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter/CG_CYLINDER_SET/height-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter/CG_CYLINDER_SET/radii-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter/CG_ELLIPSOID_SET-group*

- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter/CG_ELLIPSOID_SET/cardinality-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter/CG_ELLIPSOID_SET/center-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter/CG_ELLIPSOID_SET/half_axes_radii-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter/CG_HEXAHEDRON_SET-group*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter/CG_HEXAHEDRON_SET/cardinality-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter/CG_HEXAHEDRON_SET/hexahedra-group*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter/CG_HEXAHEDRON_SET/hexahedra/vertices-field*
- */NXapm_paraprobe_config_selector/ENTRY/roi_selection/spatial_filter/windowing_method-field*
- */NXapm_paraprobe_config_selector/ENTRY/time_stamp-field*
- */NXapm_paraprobe_config_selector/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_config_selector.nxdl.xml

NXapm_paraprobe_config_spatstat

Status:

application definition, extends *NXobject*

Description:

Configuration of a paraprobe-spatstat tool run in atom probe microscopy.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ivecmax: Maximum number of atoms per molecular ion. Should be 32 for paraprobe.

n_ion_source: Number of different sources iontypes to distinguish.

n_ion_target: Number of different target iontypes to distinguish.

Groups cited:
NXapm_input_ranging, *NXapm_input_reconstruction*, *NXcg_cylinder_set*, *NXcg_ellipsoid_set*, *NXcg_face_list_data_structure*, *NXcg_hexahedron_set*, *NXcs_filter_boolean_mask*, *NXcs_prng*, *NXcs_profiling*, *NXentry*, *NXmatch_filter*, *NXprocess*, *NXspatial_filter*, *NXsubsampling_filter*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_config_spatstat`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

time_stamp: (required) *NX_DATE_TIME*

ISO 8601 formatted time code with local time zone offset to UTC information included when this configuration file was created.

number_of_processes: (required) *NX_UINT* {units=*NX_UNITLESS*}

How many range_with_existent_iontypes processes should the tool execute as part of the analysis.

PROCESS: (required) *NXprocess <=*

spatial_statistics: (optional) *NXprocess*

randomize_ion_types: (required) *NX_BOOLEAN*

Specifies if the iontypes are randomized for the point cloud or not. Internally paraprobe uses a sequentially executed deterministic MT19987 (MersenneTwister) pseudo-random number generator to shuffle the iontype labels randomly across the entire set of ions.

ion_query_type_source: (required) *NX_CHAR*

How should the iontype be interpreted on the source-side, i.e. all these ion positions where a regions-of-interest (ROI) around so-called source ions will be placed. Different options exist how iontypes are interpreted given an iontype represents in general a (molecular) ion with different isotopes that have individually different multiplicity.

The value resolve_all will set an ion active in the analysis regardless of which iontype it is. Each active ion is accounted for once.

The value resolve_unknown will set an ion active when the ion is of the UNKNOWNTYPE type. Each active ion is accounted for once.

The value resolve_ion will set an ion active if it is of the specific iontype, irregardless of its elemental or isotopic details. Each active ion is counted once.

The value resolve_element will set an ion active, and most importantly, account for each as many times as the (molecular) ion contains atoms of elements in the whitelist ion_query_isotope_vector.

The value resolve_isotope will set an ion active, and most importantly, account for each as many times as the (molecular) ion contains isotopes in the whitelist ion_query_isotope_vector.

In effect, ion_query_isotope_vector acts as a whitelist to filter which ions are considered as source ions of the correlation statistics and how the multiplicity of each ion will be factorized, i.e. how often it is accounted for.

Any of these values:

- `resolve_all`
- `resolve_unknown`
- `resolve_ion`
- `resolve_element`
- `resolve_isotope`

ion_query_isotope_vector_source: (required) *NX_UINT* (Rank: 2, Dimensions: [n_ion_source, n_ivecmax]) {units=*NX_UNITLESS*}

Matrix of isotope vectors, as many as rows as different candidates for iontypes should be distinguished as possible source iontypes. In the simplest case, the matrix contains only the proton number of the element in the row, all other values set to zero. Combined with ion_query_type_source set to resolve_element this will recover usual spatial correlation statistics like the 1NN C-C spatial statistics.

ion_query_type_target: (required) *NX_CHAR*

Similarly as ion_query_type_source how should iontypes be interpreted on the target-side, i.e. how many counts will be bookkept for ions which are neighbors of source ions within or on the surface of each inspection/ROI about each source ion. Source ion in the center of the ROI are not accounted for during counting the summary statistics. For details about the resolve values consider the explanations in ion_query_type_source. These account for ion_query_type_target as well.

Any of these values:

- `resolve_all`
- `resolve_unknown`
- `resolve_ion`
- `resolve_element`
- `resolve_isotope`

ion_query_isotope_vector_target: (required) *NX_UINT* (Rank: 2, Dimensions: [n_ion_target, n_ivecmax]) {units=*NX_UNITLESS*}

Matrix of isotope vectors, as many as rows as different candidates for iontypes to distinguish as possible targets. See additional comments under ion_query_isotope_vector_source.

dataset: (required) *NXapm_input_reconstruction*

filename: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

dataset_name_reconstruction: (required) *NX_CHAR <=*

dataset_name_mass_to_charge: (required) *NX_CHAR <=*

iontypes: (required) *NXapm_input_ranging*

filename: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

group_name_iontypes: (required) *NX_CHAR <=*

spatial_filter: (optional) *NXspatial_filter*

windowing_method: (required) *NX_CHAR <=*

CG_ELLIPSOID_SET: (optional) *NXcg_ellipsoid_set <=*

dimensionality: (required) *NX_POSINT <=*

cardinality: (required) *NX_POSINT <=*

identifier_offset: (required) *NX_INT <=*

center: (required) *NX_NUMBER <=*

half_axes_radii: (required) *NX_NUMBER <=*

orientation: (required) *NX_NUMBER <=*

CG_CYLINDER_SET: (optional) *NXcg_cylinder_set <=*

dimensionality: (required) *NX_POSINT <=*

cardinality: (required) *NX_POSINT <=*

identifier_offset: (required) *NX_INT <=*

center: (required) *NX_NUMBER <=*

height: (required) *NX_NUMBER <=*

radii: (required) *NX_NUMBER <=*

CG_HEXAHEDRON_SET: (optional) *NXcg_hexahedron_set <=*

dimensionality: (required) *NX_POSINT <=*

cardinality: (required) *NX_POSINT <=*

identifier_offset: (required) *NX_INT <=*

hexahedra: (required) *NXcg_face_list_data_structure <=*

CS_FILTER_BOOLEAN_MASK: (optional) *NXcs_filter_boolean_mask <=*

number_of_objects: (required) *NX_UINT* <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

identifier: (required) *NX_UINT* <=

evaporation_id_filter: (optional) *NXsubsampling_filter*

iontype_filter: (optional) *NXmatch_filter*

hit_multiplicity_filter: (optional) *NXmatch_filter*

ion_to_edge_distances: (optional) *NXprocess*

The tool enables to inject precomputed distances of each ion to a representation of the edge of the dataset which can be used to control and substantially reduce edge effects when computing spatial statistics. **filename**: (required) *NX_CHAR*

Name of an HDF5 file which contains ion-to-edge distances.

dataset_name_distances: (required) *NX_CHAR*

Absolute HDF5 path to the dataset with the ion-to-edge distance values for each ion. The shape of the distance values has to match the length of the ion positions array in dataset/dataset_name_reconstruction and dataset_name_mass_to_charge respectively.

edge_distance: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Threshold to define how large an ion has to lay at least far away from the edge of the dataset so that the ion can act as a source, i.e. that an ROI is placed at the location of the ion and its neighbors are analyzed how they contribute to the computed statistics.

The ion_to_edge_distances threshold can be combined with a threshold for the ion_to_feature_distances. Specifically, if ion_to_feature_distances are loaded an ion only acts as a source if both threshold criteria are met.

The threshold is useful to process the dataset such that ROIs do not protrude out of the dataset as this would add bias.

ion_to_feature_distances: (optional) *NXprocess*

In addition to spatial filtering, and considering how far ions lie to the edge of the dataset, it is possible to restrict the analyses to a sub-set of ions within a distance not farther away to a feature than a threshold value. **filename**: (required) *NX_CHAR*

Name of an HDF5 file which contains ion-to-feature distances.

dataset_name_distances: (required) *NX_CHAR*

Absolute HDF5 path to the dataset with the ion-to-feature distance values for each ion.

threshold: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Threshold to define how close an ion has to lay to a feature so that the ion can at all qualify as a source, i.e. that an ROI is placed at the location of the ion and its neighbors are then analyzed how they contribute to the computed statistics.

Recall that the ion_to_feature_distances threshold is combined with the ion_to_edge_distances threshold.

random_number_generator: (recommended) *NXcs_prng*

type: (required) *NX_CHAR <=*

seed: (required) *NX_NUMBER <=*

warmup: (required) *NX_NUMBER <=*

statistics: (required) *NXprocess*

Specifies which spatial statistics to compute.

knn: (optional) *NXprocess*

Compute k-th nearest neighbour statistics.

nth: (required) *NX_UINT* {units=*NX_UNITLESS*}

Order k.

histogram_min_incr_max: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

Minimum value, increment, and maximum value of the histogram binning.

rdf: (optional) *NXprocess*

Compute radial distribution function.

histogram_min_incr_max: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

Minimum value, increment, and maximum value of the histogram binning.

performance: (required) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR <=*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_config_spatstat/ENTRY-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/analysis_description-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/definition-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/number_of_processes-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/performance-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics-group*

- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/dataset-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/dataset/dataset_name_mass_to_charge-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/dataset/dataset_name_reconstruction-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/dataset/filename-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/dataset/filename@version-attribute*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/evaporation_id_filter-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/hit_multiplicity_filter-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/ion_query_isotope_vector_source-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/ion_query_isotope_vector_target-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/ion_query_type_source-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/ion_query_type_target-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/ion_to_edge_distances-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/ion_to_edge_distances/dataset_name_distances-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/ion_to_edge_distances/edge_distance-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/ion_to_edge_distances/filename-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/ion_to_feature_distances-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/ion_to_feature_distances/dataset_name_distances-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/ion_to_feature_distances/filename-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/ion_to_feature_distances/threshold-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/iontype_filter-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/iontypes-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/iontypes/filename-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/iontypes/filename@version-attribute*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/iontypes/group_name_iontypes-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/random_number_generator-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/random_number_generator/seed-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/random_number_generator/type-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/random_number_generator/warmup-field*

- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/randomize_ion_types-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_CYLINDER_SET-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_CYLINDER_SET/cardinality-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_CYLINDER_SET/center-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_CYLINDER_SET/dimensionality-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_CYLINDER_SET/height-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_CYLINDER_SET/identifier_offset-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_CYLINDER_SET/radii-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_ELLIPSOID_SET-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_ELLIPSOID_SET/cardinality-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_ELLIPSOID_SET/center-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_ELLIPSOID_SET/dimensionality-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_ELLIPSOID_SET/half_axes_radii-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_ELLIPSOID_SET/identifier_offset-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_ELLIPSOID_SET/orientation-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_HEXAHEDRON_SET-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_HEXAHEDRON_SET/cardinality-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_HEXAHEDRON_SET/dimensionality field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_HEXAHEDRON_SET/hexahedra-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CG_HEXAHEDRON_SET/identifier_offse field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CS_FILTER_BOOLEAN_MASK-group*

- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CS_FILTER_BOOLEAN_MASK/bitdepth-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CS_FILTER_BOOLEAN_MASK/identifier field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CS_FILTER_BOOLEAN_MASK/mask-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/CS_FILTER_BOOLEAN_MASK/number_ field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/spatial_filter/windowing_method-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/statistics-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/statistics/knn-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/statistics/knn/histogram_min_incr_max-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/statistics/knn/nth-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/statistics/rdf-group*
- */NXapm_paraprobe_config_spatstat/ENTRY/PROCESS/spatial_statistics/statistics/rdf/histogram_min_incr_max-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/program-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/program@version-attribute*
- */NXapm_paraprobe_config_spatstat/ENTRY/results_path-field*
- */NXapm_paraprobe_config_spatstat/ENTRY/time_stamp-field*
- */NXapm_paraprobe_config_spatstat/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_config_spatstat.nxdl.xml

NXapm_paraprobe_config_surfacer

Status:

application definition, extends *NXobject*

Description:

Configuration of a paraprobe-surfacer tool run in atom probe microscopy.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_alpha_values: Number of alpha values (and offset values) to probe.

n_values: How many different match values does the filter specify.

Groups cited:
NXapm_input_ranging, *NXapm_input_reconstruction*, *NXcg_cylinder_set*, *NXcg_ellipsoid_set*, *NXcg_face_list_data_structure*, *NXcg_hexahedron_set*, *NXcs_filter_boolean_mask*, *NXcs_profiling*, *NXentry*, *NXmatch_filter*, *NXprocess*, *NXspatial_filter*, *NXsubsampling_filter*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_config_surfacer`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

time_stamp: (required) *NX_DATE_TIME*

ISO 8601 formatted time code with local time zone offset to UTC information included when this configuration file was created.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

number_of_processes: (required) *NX_UINT* {units=*NX_UNITLESS*}

For now a support field for the tool to identify how many individual analyses the tool should executed as part of the analysis.

PROCESS: (required) *NXprocess* <=

dataset: (required) *NXapm_input_reconstruction*

filename: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

dataset_name_reconstruction: (required) *NX_CHAR* <=

dataset_name_mass_to_charge: (required) *NX_CHAR* <=

iontypes: (required) *NXapm_input_ranging*

filename: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

group_name_iontypes: (required) *NX_CHAR* <=

spatial_filter: (optional) *NXspatial_filter*

windowing_method: (required) *NX_CHAR* <=

CG_ELLIPSOID_SET: (optional) *NXcg_ellipsoid_set* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

identifier_offset: (required) *NX_INT* <=

center: (required) *NX_NUMBER* <=

half_axes_radii: (required) *NX_NUMBER* <=

orientation: (required) *NX_NUMBER* <=

CG_CYLINDER_SET: (optional) *NXcg_cylinder_set* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

identifier_offset: (required) *NX_INT* <=

center: (required) *NX_NUMBER* <=

height: (required) *NX_NUMBER* <=

radii: (required) *NX_NUMBER* <=

CG_HEXAHEDRON_SET: (optional) *NXcg_hexahedron_set* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

identifier_offset: (required) *NX_INT* <=

hexahedra: (required) *NXcg_face_list_data_structure* <=

CS_FILTER_BOOLEAN_MASK: (optional) *NXcs_filter_boolean_mask* <=

number_of_objects: (required) *NX_UINT* <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

identifier: (required) *NX_UINT* <=

evaporation_id_filter: (optional) *NXsubsampling_filter*

linear_range_min_incr_max: (required) *NX_UINT* (Rank: 1, Dimensions: [3]) {units=*NX_UNITLESS*} <=

iontype_filter: (optional) *NXmatch_filter*

method: (required) *NX_CHAR* <=

match: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_values]) {units=*NX_UNITLESS*} <=

hit_multiplicity_filter: (optional) *NXmatch_filter*

method: (required) *NX_CHAR* <=

match: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_values]) {units=*NX_UNITLESS*} <=

surface_meshing: (required) *NXprocess*

preprocessing_method: (required) *NX_CHAR*

Specifies the method that is used to preprocess the point cloud. The main purpose of this setting is to specify whether the point cloud should be segmented or not during the preprocessing to identify which points are more likely lying close to the edge of the point cloud. These points could be more relevant than the interior points for certain alpha-shape constructions.

By default no such filtering is used during pre-processing. By contrast, the option kuehbach activates a preprocessing during which a Hoshen-Kopelman percolation analysis is used to identify which points are closer to the edge of the dataset. This can reduce the number of points in the alpha-shape computation and thus improve performance substantially. Details about the methods are reported in M. Kühbach et al..

Any of these values: `default` | `kuehbach`

preprocessing_kernel_width: (required) *NX_UINT* {units=*NX_UNITLESS*}

When using the kuehbach preprocessing, this is the width of the kernel for identifying which ions are in voxels close to the edge of the point cloud.

alpha_value_choice: (required) *NX_CHAR*

Specifies which method to use to define the alpha value. The value convex_hull_naive is the default. This instructs the tool to use a fast specialized algorithm for computing only the convex hull. The resulting triangles can be skinny.

The value convex_hull_refine computes first also a convex_hull_naive but refines the mesh by triangle flipping and splitting to improve the quality of the mesh.

The value smallest_solid instructs the CGAL library to choose a value which realizes an alpha-shape that is the smallest solid.

The value cgal_optimal instructs the library to choose a value which the library considers as an optimal value. Details are define in the respective section of the CGAL library on 3D alpha shapes.

The value set_of_values instructs to compute a list of alpha-shapes for the specified alpha-values.

The value set_of_alpha_wrappings instructs the library to generate a set of so-called alpha wrappings. These are a method which is similar to alpha shapes but provide additional guarantees though such as watertightness and proximity constraints on the resulting wrapping.

Any of these values:

- `convex_hull_naive`
- `convex_hull_refine`
- `smallest_solid`
- `cgal_optimal`

- `set_of_values`
- `set_of_alpha_wrappings`

alpha_values: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_alpha_values]) {units=*NX_ANY*}

Array of alpha values to use when alpha_value_choice is set_of_values or when alpha_value_choice is set_of_alpha_wrappings.

offset_values: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_alpha_values]) {units=*NX_LENGTH*}

Array of offset values to use when alpha_value_choice is set_of_alpha_wrappings. The array of alpha_values and offset_values define a sequence of (alpha and offset value).

has_exterior_facets: (required) *NX_BOOLEAN*

Specifies if the tool should compute the set of exterior triangle facets for each alpha complex (for convex hull, alpha shapes, and wrappings)

has_closure: (required) *NX_BOOLEAN*

Specifies if the tool should check if the alpha complex of exterior triangular facets is a closed polyhedron.

has_interior_tetrahedra: (required) *NX_BOOLEAN*

Specifies if the tool should compute all interior tetrahedra of the alpha complex (currently only for alpha shapes).

performance: (required) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* *<=*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_config_surfacer/ENTRY-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/analysis_description-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/definition-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/number_of_processes-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/performance-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/dataset-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/dataset/dataset_name_mass_to_charge-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/dataset/dataset_name_reconstruction-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/dataset/filename-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/dataset/filename@version-attribute*

- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/evaporation_id_filter-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/evaporation_id_filter/linear_range_min_incr_max-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/hit_multiplicity_filter-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/hit_multiplicity_filter/match-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/hit_multiplicity_filter/method-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/iontype_filter-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/iontype_filter/match-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/iontype_filter/method-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/iontypes-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/iontypes/filename-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/iontypes/filename@version-attribute*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/iontypes/group_name_iontypes-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/cardinality-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/center-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/dimensionality-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/height-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/identifier_offset-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/radii-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/cardinality-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/center-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/dimensionality-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/half_axes_radii-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/identifier_offset-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/orientation-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/cardinality-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/dimensionality-field*

- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/hexahedra-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/identifier_offset-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/bitdepth-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/identifier-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/mask-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/number_of_objects-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/spatial_filter/windowing_method-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/surface_meshing-group*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/surface_meshing/alpha_value_choice-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/surface_meshing/alpha_values-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/surface_meshing/has_closure-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/surface_meshing/has_exterior_facets-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/surface_meshing/has_interior_tetrahedra-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/surface_meshing/offset_values-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/surface_meshing/preprocessing_kernel_width-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/PROCESS/surface_meshing/preprocessing_method-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/program-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/program@version-attribute*
- */NXapm_paraprobe_config_surfacer/ENTRY/results_path-field*
- */NXapm_paraprobe_config_surfacer/ENTRY/time_stamp-field*
- */NXapm_paraprobe_config_surfacer/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_config_surfacer.nxdl.xml

NXapm_paraprobe_config_tessellator

Status:

application definition, extends *NXobject*

Description:

Configuration of a paraprobe-tessellator tool run in atom probe microscopy.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXapm_input_ranging, *NXapm_input_reconstruction*, *NXcg_cylinder_set*, *NXcg_ellipsoid_set*, *NXcg_face_list_data_structure*, *NXcg_hexahedron_set*, *NXcs_filter_boolean_mask*, *NXcs_profiling*, *NXentry*, *NXmatch_filter*, *NXprocess*, *NXspatial_filter*, *NXsubsampling_filter*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_config_tessellator`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

time_stamp: (required) *NX_DATE_TIME*

ISO 8601 formatted time code with local time zone offset to UTC information included when this configuration file was created.

number_of_processes: (required) *NX_UINT* {units=*NX_UNITLESS*}

How many individual analyses should the tool execute.

PROCESS: (required) *NXprocess <=*

dataset: (required) *NXapm_input_reconstruction*

filename: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

dataset_name_reconstruction: (required) *NX_CHAR <=*

dataset_name_mass_to_charge: (required) *NX_CHAR <=*

iontypes: (required) *NXapm_input_ranging*

filename: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR* <=

group_name_iontypes: (required) *NX_CHAR* <=

spatial_filter: (optional) *NXspatial_filter*

windowing_method: (required) *NX_CHAR* <=

CG_ELLIPSOID_SET: (optional) *NXcg_ellipsoid_set* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

identifier_offset: (required) *NX_INT* <=

center: (required) *NX_NUMBER* <=

half_axes_radii: (required) *NX_NUMBER* <=

orientation: (required) *NX_NUMBER* <=

CG_CYLINDER_SET: (optional) *NXcg_cylinder_set* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

identifier_offset: (required) *NX_INT* <=

center: (required) *NX_NUMBER* <=

height: (required) *NX_NUMBER* <=

radii: (required) *NX_NUMBER* <=

CG_HEXAHEDRON_SET: (optional) *NXcg_hexahedron_set* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

identifier_offset: (required) *NX_INT* <=

hexahedra: (required) *NXcg_face_list_data_structure* <=

CS_FILTER_BOOLEAN_MASK: (optional) *NXcs_filter_boolean_mask* <=

number_of_objects: (required) *NX_UINT* <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

identifier: (required) *NX_UINT* <=

evaporation_id_filter: (optional) *NXsubsampling_filter*

iontype_filter: (optional) *NXmatch_filter*

hit_multiplicity_filter: (optional) *NXmatch_filter*

ion_to_edge_distances: (optional) *NXprocess*

The tool enables to inject precomputed distance information for each point which can be used for further post-processing and analysis. **filename**: (required) *NX_CHAR*

Name of an HDF5 file which contains the ion distances. Users are responsible this file and referred to dataset under dataset_name have an ion_distance value for each ion. **@version**: (required) *NX_CHAR*

Version identifier of the file such as a secure hash which documents the binary state of the file to add an additional layer of reproducibility.

dataset_name: (required) *NX_CHAR*

Absolute HDF5 path to the dataset with distance values for each ion.

tessellating: (required) *NXprocess*

method: (required) *NX_CHAR*

Specifies for which points the tool will compute the tessellation. By default, a Voronoi tessellation is computed for all ions in the filtered point cloud.

Obligatory value: `default`

has_cell_volume: (required) *NX_BOOLEAN*

Specifies if the tool should report the volume of each cell.

has_cell_neighbors: (required) *NX_BOOLEAN*

Specifies if the tool should report the first-order neighbors of each cell.

has_cell_geometry: (required) *NX_BOOLEAN*

Specifies if the tool should report the facets and vertices of each cell.

has_cell_edge_detection: (required) *NX_BOOLEAN*

Specifies if the tool should report if the cell makes contact with the tight axis-aligned bounding box about the point cloud. This can be used to identify if the shape of the cell is affected by the edge of the dataset or if cells are deeply enough embedded into the point cloud so that the shape of their cells are not affected by the presence of the boundary.

performance: (required) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* *<=*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_config_tessellator/ENTRY-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/analysis_description-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/definition-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/number_of_processes-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/performance-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS-group*

- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/dataset-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/dataset/dataset_name_mass_to_charge-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/dataset/dataset_name_reconstruction-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/dataset/filename-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/dataset/filename@version-attribute*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/evaporation_id_filter-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/hit_multiplicity_filter-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/ion_to_edge_distances-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/ion_to_edge_distances/dataset_name-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/ion_to_edge_distances/filename-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/ion_to_edge_distances/filename@version-attribute*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/iontype_filter-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/iontypes-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/iontypes/filename-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/iontypes/filename@version-attribute*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/iontypes/group_name_iontypes-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/cardinality-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/center-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/dimensionality-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/height-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/identifier_offset-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_CYLINDER_SET/radii-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/cardinality-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/center-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/dimensionality-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/half_axes_radii-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/identifier_offset-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_ELLIPSOID_SET/orientation-field*

- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/cardinality-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/dimensionality-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/hexahedra-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CG_HEXAHEDRON_SET/identifier_offset-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/bitdepth-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/identifier-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/mask-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/CS_FILTER_BOOLEAN_MASK/number_of_objects-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/spatial_filter/windowing_method-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/tessellating-group*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/tessellating/has_cell_edge_detection-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/tessellating/has_cell_geometry-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/tessellating/has_cell_neighbors-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/tessellating/has_cell_volume-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/PROCESS/tessellating/method-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/program-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/program@version-attribute*
- */NXapm_paraprobe_config_tessellator/ENTRY/results_path-field*
- */NXapm_paraprobe_config_tessellator/ENTRY/time_stamp-field*
- */NXapm_paraprobe_config_tessellator/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_config_tessellator.nxdl.xml

NXapm_paraprobe_config_transcoder

Status:

application definition, extends *NXobject*

Description:

Configurations of a paraprobe-transcoder tool run in atom probe microscopy.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXapm_input_ranging, *NXapm_input_reconstruction*, *NXcs_profiling*, *NXentry*, *NXprocess*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_config_transcoder`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ideally an ever persistent resource where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable deterministically.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

time_stamp: (required) *NX_DATE_TIME*

ISO 8601 formatted time code with local time zone offset to UTC information included when this configuration file was created.

PROCESS: (required) *NXprocess* <=

dataset: (required) *NXapm_input_reconstruction*

filename: (required) *NX_CHAR* <=

The path and name of the file (technology partner or community format) from which to read the reconstructed ion positions. Currently, POS, ePOS, APT files from APSuite, and NXS, i.e. NeXus/HDF5 files are supported. **@version**: (required) *NX_CHAR* <=

iontypes: (required) *NXapm_input_ranging*

filename: (required) *NX_CHAR* <=

The path and name of the file (technology partner or community format from which to read the ranging definitions, i.e. how to map mass-to-charge-state ratios on iontypes. Currently, RRNG, RNG, and NXS, i.e. NeXus/HDF5 files are supported. **@version**: (required) *NX_CHAR* <=

performance: (required) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_config_transcoder/ENTRY-group*
- */NXapm_paraprobe_config_transcoder/ENTRY/analysis_description-field*
- */NXapm_paraprobe_config_transcoder/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_config_transcoder/ENTRY/definition-field*
- */NXapm_paraprobe_config_transcoder/ENTRY/performance-group*
- */NXapm_paraprobe_config_transcoder/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_config_transcoder/ENTRY/PROCESS-group*
- */NXapm_paraprobe_config_transcoder/ENTRY/PROCESS/dataset-group*
- */NXapm_paraprobe_config_transcoder/ENTRY/PROCESS/dataset/filename-field*
- */NXapm_paraprobe_config_transcoder/ENTRY/PROCESS/dataset/filename@version-attribute*
- */NXapm_paraprobe_config_transcoder/ENTRY/PROCESS/iontypes-group*
- */NXapm_paraprobe_config_transcoder/ENTRY/PROCESS/iontypes/filename-field*
- */NXapm_paraprobe_config_transcoder/ENTRY/PROCESS/iontypes/filename@version-attribute*
- */NXapm_paraprobe_config_transcoder/ENTRY/program-field*
- */NXapm_paraprobe_config_transcoder/ENTRY/program@version-attribute*
- */NXapm_paraprobe_config_transcoder/ENTRY/results_path-field*
- */NXapm_paraprobe_config_transcoder/ENTRY/time_stamp-field*
- */NXapm_paraprobe_config_transcoder/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_config_transcoder.nxdl.xml

NXapm_paraprobe_results_clusterer

Status:

application definition, extends *NXobject*

Description:

Results of a paraprobe-clusterer tool run.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ions: The total number of ions in the reconstruction.

n_dict: The total number of entries in the restricted_identifier dictionary.

Groups cited:

NXcoordinate_system_set, *NXcs_computer*, *NXcs_cpu*, *NXcs_filter_boolean_mask*, *NXcs_gpu*, *NXcs_io_obj*, *NXcs_io_sys*, *NXcs_mm_sys*, *NXcs_profiling_event*, *NXcs_profiling*, *NXentry*, *NXfabrication*, *NXprocess*, *NXsimilarity_grouping*, *NXtransformations*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_results_clusterer`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

start_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file was started, i.e. the paraprobe-tool executable started as a process.

end_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file were completed and the paraprobe-tool executable exited as a process.

config_filename: (required) *NX_CHAR*

The absolute path and name of the config file for this analysis.

@version: (required) *NX_CHAR*

At least SHA256 strong hash of the specific config_file for tracking provenance.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

status: (required) *NX_CHAR*

A statement whether the paraprobe-tool executable managed to process the analysis or failed prematurely.

This status is written to the results file after the end_time at which point the executable must no longer compute analyses. Only when this status message is present and shows *success*, the user should consider the results. In all other cases, it might be that the executable has terminated prematurely or another error occurred.

Any of these values: `success` | `failure`

USER: (optional) *NXuser <=*

If used, contact information and eventually details of at least the person who performed this analysis. **name**: (required) *NX_CHAR <=*

affiliation: (recommended) *NX_CHAR <=*

address: (optional) *NX_CHAR <=*

email: (recommended) *NX_CHAR <=*

orcid: (recommended) *NX_CHAR <=*

orcid_platform: (recommended) *NX_CHAR <=*

telephone_number: (optional) *NX_CHAR <=*

role: (recommended) *NX_CHAR <=*

social_media_name: (optional) *NX_CHAR <=*

social_media_platform: (optional) *NX_CHAR <=*

COORDINATE_SYSTEM_SET: (required) *NXcoordinate_system_set*

Details about the coordinate system conventions used. If nothing else is specified we assume that there has to be at least one set of NXtransformations named paraprobe defined, which specifies the coordinate system. In which all positions are defined. **TRANSFORMATIONS**: (required) *NXtransformations <=*

The individual coordinate systems which should be used. Field names should be prefixed with the following controlled terms indicating which individual coordinate system is described:

- paraprobe
- lab
- specimen
- laser

- leap
- detector
- recon

PROCESS: (optional) *NXprocess* *<=*

window: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies which of the ions in the dataset were analyzed during this process. **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of ions covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} *<=*

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*} *<=*

The unsigned integer array representing the content of the mask. If padding is used, padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe-toolbox executable. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth (padding).

cluster_analysis: (optional) *NXprocess*

The result of a cluster analyses. These include typically the label for each ion/point documenting to which feature (if any) an ion is assigned. Typically, each analysis/run yields only a single cluster. In cases of fuzzy clustering it can be possible that an ion is assigned to multiple cluster (eventually with different) weight/probability. **dbscanID**: (optional) *NXsimilarity_grouping*

Results of a DBScan clustering analysis.

eps: (required) *NX_FLOAT* {units=*NX_LENGTH*}

The epsilon (eps) parameter.

min_pts: (required) *NX_UINT* {units=*NX_UNITLESS*}

The minimum points (min_pts) parameter.

cardinality: (required) *NX_UINT* {units=*NX_UNITLESS*} *<=*

Number of members in the set which is partitioned into features. Specifically, this is the total number of targets filtered from the dataset. Cardinality here is not the total number of ions in the dataset.

identifier_offset: (required) *NX_NUMBER* {units=*NX_UNITLESS*}

Which identifier is the first to be used to label a cluster.

The value should be chosen in such a way that special values can be resolved: * identifier_offset-1 indicates an object belongs to no cluster. * identifier_offset-2 indicates an object belongs to the noise category. Setting for instance identifier_offset to 1 recovers the commonly used case that objects of the noise category get values to -1 and unassigned points to 0. Numerical identifier have to be strictly increasing.

targets: (required) *NX_UINT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

The evaporation sequence identifier to figure out which ions from the reconstruction were considered targets.

model_labels: (required) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

The raw labels from the DBScan clustering backend process.

core_sample_indices: (required) *NX_INT* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

The raw array of core sample indices which specify which of the targets are core points.

numerical_label: (required) *NX_UINT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*} *<=*

Matrix of numerical label for each member in the set. For classical clustering algorithms this can for instance encode the cluster_identifier.

weight: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

The array of weight which specifies how surely/likely the cluster is associated/assigned to a specific identifier as is specified in the cluster_identifier array. For the DBScan and atom probe tomography the multiplicity of each ion with respect to the cluster. That is how many times should the position of the ion be accounted for because the ion is e.g. a molecular ion with several elements or isotope of requested type.

is_noise: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [c])

Optional bitmask encoding if members of the set are assigned to as noise or not.

is_core: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [c])

Optional bitmask encoding if member of the set are a core point. For details to which feature/cluster an ion/point is a core point consider numerical_label.

statistics: (required) *NXprocess* *<=*

In addition to the detailed storage which members was grouped to which feature/group summary statistics are stored under this group. **number_of_noise**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Total number of members in the set which are categorized as noise.

number_of_core: (required) *NX_UINT* {units=*NX_UNITLESS*}

Total number of members in the set which are categorized as a core point.

number_of_features: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Total number of clusters (excluding noise and unassigned).

feature_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [n_features]) {units=*NX_UNITLESS*} <=

Array of numerical identifier of each feature (cluster).

feature_member_count: (required) *NX_UINT* (Rank: 1, Dimensions: [n_features]) {units=*NX_UNITLESS*} <=

Array of number of members for each feature.

performance: (recommended) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* <=

command_line_call: (optional) *NX_CHAR* <=

start_time: (recommended) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

total_elapsed_time: (required) *NX_NUMBER* <=

number_of_processes: (required) *NX_POSINT* <=

number_of_threads: (required) *NX_POSINT* <=

number_of_gpus: (required) *NX_POSINT* <=

CS_COMPUTER: (recommended) *NXcs_computer* <=

name: (recommended) *NX_CHAR* <=

operating_system: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

uuid: (optional) *NX_CHAR* <=

CS_CPU: (optional) *NXcs_cpu* <=

name: (optional) *NX_CHAR* <=

FABRICATION: (recommended) *NXfabrication* <=

identifier: (optional) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CS_GPU: (optional) *NXcs_gpu* <=

name: (optional) *NX_CHAR* <=

FABRICATION: (recommended) *NXfabrication* <=

identifier: (optional) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CS_MM_SYS: (optional) *NXcs_mm_sys <=*

total_physical_memory: (required) *NX_NUMBER <=*

CS_IO_SYS: (optional) *NXcs_io_sys <=*

CS_IO_OBJ: (required) *NXcs_io_obj <=*

technology: (required) *NX_CHAR <=*

max_physical_capacity: (required) *NX_NUMBER <=*

name: (optional) *NX_CHAR <=*

FABRICATION: (recommended) *NXfabrication <=*

identifier: (optional) *NX_CHAR <=*

capabilities: (optional) *NX_CHAR*

CS_PROFILING_EVENT: (required) *NXcs_profiling_event*

start_time: (optional) *NX_DATE_TIME <=*

end_time: (optional) *NX_DATE_TIME <=*

description: (required) *NX_CHAR <=*

elapsed_time: (required) *NX_NUMBER <=*

number_of_processes: (required) *NX_POSINT <=*

Specify if it was different from the number_of_processes in the NXcs_profiling super class.

number_of_threads: (required) *NX_POSINT <=*

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

number_of_gpus: (required) *NX_POSINT <=*

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

max_virtual_memory_snapshot: (recommended) *NX_NUMBER <=*

max_resident_memory_snapshot: (recommended) *NX_NUMBER <=*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_results_clusterer/ENTRY-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/analysis_description-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/config_filename-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/config_filename@version-attribute*
- */NXapm_paraprobe_results_clusterer/ENTRY/COORDINATE_SYSTEM_SET-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-group*

- */NXapm_paraprobe_results_clusterer/ENTRY/definition-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/end_time-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/command_line_call-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_CPU-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_CPU/name-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_GPU-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_GPU/name-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/capabilit field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identifier field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/max_physical_capacity-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/name-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/technology-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_MM_SYS-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_MM_SYS/total_physical_memory-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/description-field*

- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/elapsed_time-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/end_time-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_resident_memory_s field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_sn field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_gpus-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_processes-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_threads-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/start_time-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/name-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/operating_system-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/operating_system@version-attribute*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/uuid-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/end_time-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/number_of_gpus-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/number_of_processes-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/number_of_threads-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/start_time-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/performance/total_elapsed_time-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/cardinality-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/core_sample_indices-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/eps-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/identifier_offset-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/is_core-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/is_noise-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/min_pts-field*

- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/model_labels-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/numerical_label-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/statistics-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/statistics/feature_identifier-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/statistics/feature_member_count-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/statistics/number_of_core-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/statistics/number_of_features-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/statistics/number_of_noise-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/targets-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/cluster_analysis/dbscanID/weight-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/window-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/window/bitdepth-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/window/mask-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/PROCESS/window/number_of_ions-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/program-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/program@version-attribute*
- */NXapm_paraprobe_results_clusterer/ENTRY/results_path-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/start_time-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/status-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/USER-group*
- */NXapm_paraprobe_results_clusterer/ENTRY/USER/address-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/USER/affiliation-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/USER/email-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/USER/name-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/USER/orcid-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/USER/orcid_platform-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/USER/role-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/USER/social_media_name-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/USER/social_media_platform-field*
- */NXapm_paraprobe_results_clusterer/ENTRY/USER/telephone_number-field*
- */NXapm_paraprobe_results_clusterer/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_results_clusterer.nxdl.xml

NXapm_paraprobe_results_distancer

Status:

application definition, extends *NXobject*

Description:

Results of a paraprobe-distancer tool run.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ions: The total number of ions in the reconstruction.

n_tris: The total number of triangles in the set.

Groups cited:
NXcoordinate_system_set, *NXcs_computer*, *NXcs_cpu*, *NXcs_filter_boolean_mask*, *NXcs_gpu*, *NXcs_io_obj*, *NXcs_io_sys*, *NXcs_mm_sys*, *NXcs_profiling_event*, *NXcs_profiling*, *NXentry*, *NXfabrication*, *NXprocess*, *NXtransformations*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_results_distancer`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

start_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file was started, i.e. the paraprobe-tool executable started as a process.

end_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file were completed and the paraprobe-tool executable exited as a process.

config_filename: (required) *NX_CHAR*

The absolute path and name of the config file for this analysis.

@version: (required) *NX_CHAR*

At least SHA256 strong hash of the specific config_file for tracking provenance.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

status: (required) *NX_CHAR*

A statement whether the paraprobe-tool executable managed to process the analysis or failed prematurely.

This status is written to the results file after the end_time at which point the executable must not compute any analysis. Only when this status message is present and shows *success*, the user should consider the results. In all other cases it might be that the executable has terminated prematurely or another error occurred.

Any of these values: `success` | `failure`

USER: (recommended) *NXuser <=*

If used, contact information and eventually details of at least the person who performed this analysis. **name**: (required) *NX_CHAR <=*

affiliation: (recommended) *NX_CHAR <=*

address: (optional) *NX_CHAR <=*

email: (recommended) *NX_CHAR <=*

orcid: (recommended) *NX_CHAR <=*

orcid_platform: (recommended) *NX_CHAR <=*

telephone_number: (optional) *NX_CHAR <=*

role: (recommended) *NX_CHAR <=*

social_media_name: (optional) *NX_CHAR <=*

social_media_platform: (optional) *NX_CHAR <=*

COORDINATE_SYSTEM_SET: (required) *NXcoordinate_system_set*

Details about the coordinate system conventions used.

TRANSFORMATIONS: (required) *NXtransformations <=*

The individual coordinate systems which should be used. Field names should be prefixed with the following controlled terms indicating which individual coordinate system is described:

- paraprobe
- lab

- specimen
- laser
- leap
- detector
- recon

PROCESS: (optional) *NXprocess* <=

point_to_triangle_set: (required) *NXprocess*

The tool can be used to compute the analytical distance of each ion to a set of triangles. **distance**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_LENGTH*}

The closest analytical distance of the ions to their respectively closest triangle from the triangle set.

triangle_identifier: (optional) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

The identifier of the triangle that is closest for each ion.

xdmf_ion_identifier: (optional) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

A support field to visualize each ion and with this the distance informations using XDMF and e.g. Paraview.

window: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies which of the ions in the dataset were analyzed. **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of ions covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*} <=

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

window_triangles: (optional) *NXcs_filter_boolean_mask*

A bitmask which identifies which of the triangles in the set were considered. Usually these are all but sometimes users may wish to filter certain portions of the triangles out. If window_triangles is not provided it means that all triangles were taken. **number_of_triangles**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of triangles covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [n_triangles]) {units=*NX_UNITLESS*} <=

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

sign_valid: (optional) *NXcs_filter_boolean_mask*

A bitmask which identifies which of the distance values can be assumed to have a consistent sign because the closest triangle had a consistent outer unit normal defined. For points whose bit is set 0 the distance is correct but the sign is not reliable. **number_of_points**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of triangles covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} <=

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0.

performance: (recommended) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* <=

command_line_call: (optional) *NX_CHAR* <=

start_time: (recommended) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

total_elapsed_time: (required) *NX_NUMBER* <=

number_of_processes: (required) *NX_POSINT* <=

number_of_threads: (required) *NX_POSINT* <=

number_of_gpus: (required) *NX_POSINT* <=

CS_COMPUTER: (recommended) *NXcs_computer* <=

- **name**: (recommended) *NX_CHAR* <=
- **operating_system**: (required) *NX_CHAR* <=
  - **@version**: (required) *NX_CHAR* <=
- **uuid**: (optional) *NX_CHAR* <=
- **CS_CPU**: (optional) *NXcs_cpu* <=
  - **name**: (optional) *NX_CHAR* <=
  - **FABRICATION**: (recommended) *NXfabrication* <=
    - **identifier**: (optional) *NX_CHAR* <=
    - **capabilities**: (optional) *NX_CHAR*
- **CS_GPU**: (optional) *NXcs_gpu* <=
  - **name**: (optional) *NX_CHAR* <=
  - **FABRICATION**: (recommended) *NXfabrication* <=
    - **identifier**: (optional) *NX_CHAR* <=
    - **capabilities**: (optional) *NX_CHAR*
- **CS_MM_SYS**: (optional) *NXcs_mm_sys* <=
  - **total_physical_memory**: (required) *NX_NUMBER* <=
- **CS_IO_SYS**: (optional) *NXcs_io_sys* <=
  - **CS_IO_OBJ**: (required) *NXcs_io_obj* <=
    - **technology**: (required) *NX_CHAR* <=
    - **max_physical_capacity**: (required) *NX_NUMBER* <=
    - **name**: (optional) *NX_CHAR* <=
    - **FABRICATION**: (recommended) *NXfabrication* <=
      - **identifier**: (optional) *NX_CHAR* <=
      - **capabilities**: (optional) *NX_CHAR*
- **CS_PROFILING_EVENT**: (required) *NXcs_profiling_event*
  - **start_time**: (optional) *NX_DATE_TIME* <=
  - **end_time**: (optional) *NX_DATE_TIME* <=
  - **description**: (required) *NX_CHAR* <=
  - **elapsed_time**: (required) *NX_NUMBER* <=
  - **number_of_processes**: (required) *NX_POSINT* <=

    Specify if it was different from the number_of_processes in the NXcs_profiling super class.
  - **number_of_threads**: (required) *NX_POSINT* <=

    Specify if it was different from the number_of_threads in the NXcs_profiling super class.
  - **number_of_gpus**: (required) *NX_POSINT* <=

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

max_virtual_memory_snapshot: (recommended) *NX_NUMBER <=*

max_resident_memory_snapshot: (recommended) *NX_NUMBER <=*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_results_distancer/ENTRY-group*
- */NXapm_paraprobe_results_distancer/ENTRY/analysis_description-field*
- */NXapm_paraprobe_results_distancer/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_results_distancer/ENTRY/config_filename-field*
- */NXapm_paraprobe_results_distancer/ENTRY/config_filename@version-attribute*
- */NXapm_paraprobe_results_distancer/ENTRY/COORDINATE_SYSTEM_SET-group*
- */NXapm_paraprobe_results_distancer/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-group*
- */NXapm_paraprobe_results_distancer/ENTRY/definition-field*
- */NXapm_paraprobe_results_distancer/ENTRY/end_time-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance-group*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/command_line_call-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER-group*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_CPU-group*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION-group*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_CPU/name-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_GPU-group*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION-group*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_GPU/name-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS-group*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ-group*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION-group*

- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/capabili field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identifier field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/max_physical_capacity-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/name-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/technology-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_MM_SYS-group*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_MM_SYS/total_physical_memory-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT-group*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/description-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/elapsed_time-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/end_time-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_resident_memory_ field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_s field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_gpus-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_processes-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_threads-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/start_time-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/name-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/operating_system-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/operating_system@version-attribute*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/uuid-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/end_time-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/number_of_gpus-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/number_of_processes-field*

- */NXapm_paraprobe_results_distancer/ENTRY/performance/number_of_threads-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/start_time-field*
- */NXapm_paraprobe_results_distancer/ENTRY/performance/total_elapsed_time-field*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS-group*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set-group*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/distance-field*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/sign_valid-group*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/sign_valid/bitdepth-field*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/sign_valid/mask-field*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/sign_valid/number_of_points-field*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/triangle_identifier-field*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/window-group*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/window/bitdepth-field*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/window/mask-field*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/window/number_of_ions-field*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/window_triangles-group*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/window_triangles/bitdepth-field*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/window_triangles/mask-field*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/window_triangles/number_of_triangles-field*
- */NXapm_paraprobe_results_distancer/ENTRY/PROCESS/point_to_triangle_set/xdmf_ion_identifier-field*
- */NXapm_paraprobe_results_distancer/ENTRY/program-field*
- */NXapm_paraprobe_results_distancer/ENTRY/program@version-attribute*
- */NXapm_paraprobe_results_distancer/ENTRY/results_path-field*
- */NXapm_paraprobe_results_distancer/ENTRY/start_time-field*
- */NXapm_paraprobe_results_distancer/ENTRY/status-field*
- */NXapm_paraprobe_results_distancer/ENTRY/USER-group*
- */NXapm_paraprobe_results_distancer/ENTRY/USER/address-field*
- */NXapm_paraprobe_results_distancer/ENTRY/USER/affiliation-field*
- */NXapm_paraprobe_results_distancer/ENTRY/USER/email-field*
- */NXapm_paraprobe_results_distancer/ENTRY/USER/name-field*
- */NXapm_paraprobe_results_distancer/ENTRY/USER/orcid-field*
- */NXapm_paraprobe_results_distancer/ENTRY/USER/orcid_platform-field*
- */NXapm_paraprobe_results_distancer/ENTRY/USER/role-field*
- */NXapm_paraprobe_results_distancer/ENTRY/USER/social_media_name-field*

- */NXapm_paraprobe_results_distancer/ENTRY/USER/social_media_platform-field*
- */NXapm_paraprobe_results_distancer/ENTRY/USER/telephone_number-field*
- */NXapm_paraprobe_results_distancer/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_results_distancer.nxdl.xml

NXapm_paraprobe_results_intersector

Status:

application definition, extends *NXobject*

Description:

Results of a paraprobe-intersector tool run.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_c2n: The total number of links pointing from current to next.

n_n2c: The total number of links pointing from next to current.

n_features_curr: The total number of members in the current_set.

n_features_next: The total number of members in the next_set.

n_cluster: The total number of cluster found for coprecipitation analysis.

n_total: The number of rows in the table/matrix for coprecipitation stats.

Groups cited:
NXcoordinate_system_set, *NXcs_computer*, *NXcs_cpu*, *NXcs_gpu*, *NXcs_io_obj*, *NXcs_io_sys*, *NXcs_mm_sys*, *NXcs_profiling_event*, *NXcs_profiling*, *NXentry*, *NXfabrication*, *NXprocess*, *NXtransformations*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_results_intersector`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

start_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file was started, i.e. the paraprobe-tool executable started as a process.

end_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file were completed and the paraprobe-tool executable exited as a process.

config_filename: (required) *NX_CHAR*

The absolute path and name of the config file for this analysis.

@version: (required) *NX_CHAR*

At least SHA256 strong hash of the specific config_file for tracking provenance.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

USER: (recommended) *NXuser <=*

If used, contact information and eventually details of at least the person who performed this analysis. **name**: (required) *NX_CHAR <=*

affiliation: (recommended) *NX_CHAR <=*

address: (optional) *NX_CHAR <=*

email: (recommended) *NX_CHAR <=*

orcid: (recommended) *NX_CHAR <=*

orcid_platform: (recommended) *NX_CHAR <=*

telephone_number: (optional) *NX_CHAR <=*

role: (recommended) *NX_CHAR <=*

social_media_name: (optional) *NX_CHAR <=*

social_media_platform: (optional) *NX_CHAR <=*

COORDINATE_SYSTEM_SET: (optional) *NXcoordinate_system_set*

Details about the coordinate system conventions used.

TRANSFORMATIONS: (required) *NXtransformations <=*

The individual coordinate systems which should be used. Field names should be prefixed with the following controlled terms indicating which individual coordinate system is described:

- paraprobe
- lab

- specimen
- laser
- leap
- detector
- recon

PROCESS: (required) *NXprocess* <=

VOLUME_VOLUME_SPATIAL_CORRELATION: (optional) *NXprocess*

The results of an overlap/intersection analysis.

current_to_next_link: (required) *NX_UINT* (Rank: 2, Dimensions: [n_c2n, 2]) {units=*NX_UNITLESS*}

A matrix of feature_identifier which specifies which named features from the current set have directed link(s) pointing to which named feature(s) from the next set.

current_to_next_link_type: (required) *NX_UINT* (Rank: 1, Dimensions: [n_c2n]) {units=*NX_UNITLESS*}

For each link/pair in current_to_next a characterization whether the link is due to a volumetric overlap (0x00 == 0), proximity (0x01 == 1), or something else unknown (0xFF == 255).

next_to_current_link: (optional) *NX_UINT* (Rank: 2, Dimensions: [n2c, 2]) {units=*NX_UNITLESS*}

A matrix of feature_identifier which specifies which named feature(s) from the next set have directed link(s) pointing to which named feature(s) from the current set. Only if the mapping whereby the links is symmetric next_to_current maps the links in current_to_next in the opposite direction.

next_to_current_link_type: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_n2c]) {units=*NX_UNITLESS*}

For each link/pair in next_to_current a characterization whether the link is due to a volumetric overlap (0x00 == 0), proximity (0x01 == 1), or something else unknown (0xFF == 255).

intersection_volume: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [c2n]) {units=*NX_VOLUME*}

For each pair of links in current_to_next the volume of the intersection, i.e. how much volume do the two features share. If features do not intersect the volume is zero.

coprecipitation_analysis: (optional) *NXprocess*

During coprecipitation analysis the current and next set are analyzed for links in a special way. Three set comparisons are made. Members of the set in each comparison are analyzed for overlap and proximity:

The first comparison is the current_set against the current_set. The second comparison is the next_set against the next_set. The third comparison is the current_set against the next_set.

Once the (forward) links for these comparisons are ready the pair relations are analyzed with respect to which feature identifier cluster in identifier space. Thereby a logical connection (link) is established between the features in the current_set and next_set. Recall that these two set typically represent different features within an observed system for otherwise the same parameterization. Examples include two sets of e.g. precipitates with differing chemical composition that were characterized in the same material volume representing a snapshot of an e.g. microstructure at the same point in time. Researchers may have performed two analyses, one to characterize precipitates A and another one to characterize percipitates B. Coprecipitation analysis now logically connects these independent characterization results to establish spatial correlations of e.g. precipitates spatial arrangement. **current_set_feature_to_cluster**: (required) *NX_UINT* (Rank: 2, Dimensions: [n_features_curr, 2]) {units=*NX_UNITLESS*}

Matrix of feature_identifier and cluster_identifier pairs which encodes the cluster to which each feature_identifier was assigned. Here for features of the current_set.

next_set_feature_to_cluster: (required) *NX_UINT* (Rank: 2, Dimensions: [n_features_next, 2]) {units=*NX_UNITLESS*}

Matrix of feature_identifier and cluster_identifier pairs which encodes the cluster to which each feature_identifier was assigned. Here for features of the next_set.

cluster_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [n_cluster]) {units=*NX_UNITLESS*}

The identifier (names) of the cluster.

cluster_composition: (required) *NX_UINT* (Rank: 2, Dimensions: [n_cluster, 3]) {units=*NX_UNITLESS*}

Pivot table as a matrix. The first column encodes how many members from the current_set are in each cluster, one row per cluster. The second column encodes how many members from the next_set are in each cluster, in the same row per cluster respectively. The last column encodes the total number of members in the cluster.

cluster_statistics: (required) *NX_UINT* (Rank: 2, Dimensions: [n_total, 2]) {units=*NX_UNITLESS*}

Pivot table as a matrix. The first column encodes the different types of clusters based on their number of members in the sub-graph. The second column encodes how many clusters with as many members exist.

performance: (recommended) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* <=

command_line_call: (optional) *NX_CHAR* <=

start_time: (recommended) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

total_elapsed_time: (required) *NX_NUMBER* <=

number_of_processes: (required) *NX_POSINT* <=

number_of_threads: (required) *NX_POSINT* <=

number_of_gpus: (required) *NX_POSINT* <=

CS_COMPUTER: (recommended) *NXcs_computer* <=

- **name**: (recommended) *NX_CHAR* <=
- **operating_system**: (required) *NX_CHAR* <=
  - **@version**: (required) *NX_CHAR* <=
- **uuid**: (optional) *NX_CHAR* <=
- **CS_CPU**: (optional) *NXcs_cpu* <=
  - **name**: (optional) *NX_CHAR* <=
  - **FABRICATION**: (recommended) *NXfabrication* <=
    - **identifier**: (optional) *NX_CHAR* <=
    - **capabilities**: (optional) *NX_CHAR*
- **CS_GPU**: (optional) *NXcs_gpu* <=
  - **name**: (optional) *NX_CHAR* <=
  - **FABRICATION**: (recommended) *NXfabrication* <=
    - **identifier**: (optional) *NX_CHAR* <=
    - **capabilities**: (optional) *NX_CHAR*
- **CS_MM_SYS**: (optional) *NXcs_mm_sys* <=
  - **total_physical_memory**: (required) *NX_NUMBER* <=
- **CS_IO_SYS**: (optional) *NXcs_io_sys* <=
  - **CS_IO_OBJ**: (required) *NXcs_io_obj* <=
    - **technology**: (required) *NX_CHAR* <=
    - **max_physical_capacity**: (required) *NX_NUMBER* <=
    - **name**: (optional) *NX_CHAR* <=
    - **FABRICATION**: (recommended) *NXfabrication* <=
      - **identifier**: (optional) *NX_CHAR* <=
      - **capabilities**: (optional) *NX_CHAR*
- **CS_PROFILING_EVENT**: (required) *NXcs_profiling_event*
  - **start_time**: (optional) *NX_DATE_TIME* <=
  - **end_time**: (optional) *NX_DATE_TIME* <=
  - **description**: (required) *NX_CHAR* <=
  - **elapsed_time**: (required) *NX_NUMBER* <=
  - **number_of_processes**: (required) *NX_POSINT* <=

    Specify if it was different from the number_of_processes in the NXcs_profiling super class.
  - **number_of_threads**: (required) *NX_POSINT* <=

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

number_of_gpus: (required) *NX_POSINT* <=

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

max_virtual_memory_snapshot: (recommended) *NX_NUMBER* <=

max_resident_memory_snapshot: (recommended) *NX_NUMBER* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_results_intersector/ENTRY-group*
- */NXapm_paraprobe_results_intersector/ENTRY/analysis_description-field*
- */NXapm_paraprobe_results_intersector/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_results_intersector/ENTRY/config_filename-field*
- */NXapm_paraprobe_results_intersector/ENTRY/config_filename@version-attribute*
- */NXapm_paraprobe_results_intersector/ENTRY/COORDINATE_SYSTEM_SET-group*
- */NXapm_paraprobe_results_intersector/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-group*
- */NXapm_paraprobe_results_intersector/ENTRY/definition-field*
- */NXapm_paraprobe_results_intersector/ENTRY/end_time-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance-group*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/command_line_call-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER-group*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_CPU-group*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION-group*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_CPU/name-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_GPU-group*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION-group*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_GPU/name-field*

- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS-group*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ-group*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION-group*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/capabi field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identifi field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/max_physical_capacity field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/name-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/technology-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_MM_SYS-group*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_MM_SYS/total_physical_memory-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT-group*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/description-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/elapsed_time-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/end_time-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_resident_memory field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_ field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_gpus-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_processes-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_threads-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/start_time-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/name-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/operating_system-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/operating_system@version-attribute*

- /NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/uuid-field
- /NXapm_paraprobe_results_intersector/ENTRY/performance/current_working_directory-field
- /NXapm_paraprobe_results_intersector/ENTRY/performance/end_time-field
- /NXapm_paraprobe_results_intersector/ENTRY/performance/number_of_gpus-field
- /NXapm_paraprobe_results_intersector/ENTRY/performance/number_of_processes-field
- /NXapm_paraprobe_results_intersector/ENTRY/performance/number_of_threads-field
- /NXapm_paraprobe_results_intersector/ENTRY/performance/start_time-field
- /NXapm_paraprobe_results_intersector/ENTRY/performance/total_elapsed_time-field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS-group
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION-group
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/coprecipitation_an group
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/coprecipitation_an field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/coprecipitation_an field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/coprecipitation_an field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/coprecipitation_an field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/coprecipitation_an field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/current_to_next_lir field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/current_to_next_lir field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/intersection_volum field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/next_to_current_lir field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/next_to_current_lir field
- /NXapm_paraprobe_results_intersector/ENTRY/program-field
- /NXapm_paraprobe_results_intersector/ENTRY/program@version-attribute
- /NXapm_paraprobe_results_intersector/ENTRY/results_path-field
- /NXapm_paraprobe_results_intersector/ENTRY/start_time-field
- /NXapm_paraprobe_results_intersector/ENTRY/USER-group
- /NXapm_paraprobe_results_intersector/ENTRY/USER/address-field
- /NXapm_paraprobe_results_intersector/ENTRY/USER/affiliation-field

- */NXapm_paraprobe_results_intersector/ENTRY/USER/email-field*
- */NXapm_paraprobe_results_intersector/ENTRY/USER/name-field*
- */NXapm_paraprobe_results_intersector/ENTRY/USER/orcid-field*
- */NXapm_paraprobe_results_intersector/ENTRY/USER/orcid_platform-field*
- */NXapm_paraprobe_results_intersector/ENTRY/USER/role-field*
- */NXapm_paraprobe_results_intersector/ENTRY/USER/social_media_name-field*
- */NXapm_paraprobe_results_intersector/ENTRY/USER/social_media_platform-field*
- */NXapm_paraprobe_results_intersector/ENTRY/USER/telephone_number-field*
- */NXapm_paraprobe_results_intersector/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_results_intersector.nxdl.xml

NXapm_paraprobe_results_nanochem

Status:

application definition, extends *NXobject*

Description:

Results of a paraprobe-nanochem tool run.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ions: The total number of ions in the reconstruction.

n_atomic: The total number of atoms in the atomic_decomposition match filter.

n_isotopic: The total number of isotopes in the isotopic_decomposition match filter.

d: The dimensionality of the delocalization grid.

c: The cardinality/total number of cells/grid points in the delocalization grid.

n_f_tri_xdmf: The total number of XDMF values to represent all faces of triangles via XDMF.

n_feature_dict: The total number of entries in a feature dictionary.

n_speci: The total number of member distinguished when reporting composition.

Groups cited:
NXcg_face_list_data_structure, *NXcg_grid*, *NXcg_hexahedron_set*, *NXcg_marching_cubes*, *NXcg_polyhedron_set*, *NXcg_triangle_set*, *NXcg_unit_normal_set*, *NXchemical_composition*, *NXcollection*, *NXcoordinate_system_set*, *NXcs_computer*, *NXcs_cpu*, *NXcs_filter_boolean_mask*, *NXcs_gpu*, *NXcs_io_obj*, *NXcs_io_sys*, *NXcs_mm_sys*, *NXcs_profiling_event*, *NXcs_profiling*, *NXdata*, *NXdelocalization*, *NXentry*, *NXfabrication*, *NXion*, *NXisocontour*, *NXmatch_filter*, *NXprocess*, *NXtransformations*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_results_nanochem`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

start_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file was started, i.e. the paraprobe-tool executable started as a process.

end_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file were completed and the paraprobe-tool executable exited as a process.

config_filename: (required) *NX_CHAR*

The absolute path and name of the config file for this analysis.

@version: (required) *NX_CHAR*

At least SHA256 strong hash of the specific config_file for tracking provenance.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

status: (required) *NX_CHAR*

A statement whether the paraprobe-tool executable managed to process the analysis or failed prematurely.

This status is written to the results file after the end_time at which point the executable must no longer compute analyses. Only when this status message is present and shows *success*, the user should consider the results. In all other cases, it might be that the executable has terminated prematurely or another error occurred.

Any of these values: `success` | `failure`

USER: (optional) *NXuser <=*

If used, contact information and eventually details of at least the person who performed this analysis. **name**: (required) *NX_CHAR* <=

affiliation: (recommended) *NX_CHAR* <=

address: (optional) *NX_CHAR* <=

email: (recommended) *NX_CHAR* <=

orcid: (recommended) *NX_CHAR* <=

orcid_platform: (recommended) *NX_CHAR* <=

telephone_number: (optional) *NX_CHAR* <=

role: (recommended) *NX_CHAR* <=

social_media_name: (optional) *NX_CHAR* <=

social_media_platform: (optional) *NX_CHAR* <=

COORDINATE_SYSTEM_SET: (required) *NXcoordinate_system_set*

Details about the coordinate system conventions used. If nothing else is specified we assume that there has to be at least one set of NXtransformations named paraprobe defined, which specifies the coordinate system. In which all positions are defined. **TRANSFORMATIONS**: (required) *NXtransformations* <=

The individual coordinate systems which should be used. Field names should be prefixed with the following controlled terms indicating which individual coordinate system is described:

- paraprobe
- lab
- specimen
- laser
- leap
- detector
- recon

PROCESS: (optional) *NXprocess* <=

window: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies which of the ions in the dataset were analyzed during this process. **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of ions covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*} <=

The unsigned integer array representing the content of the mask. If padding is used, padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe-toolbox executable. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth (padding).

iso_surface_analysis: (optional) *NXprocess*

DELOCALIZATION: (required) *NXdelocalization*

weighting_model: (required) *NX_CHAR* <=

The weighting model specifies how mark data are mapped to a weight per point/ion. For atom probe microscopy (APM) mark data are e.g. which iontype an ion has. As an example, different models are used which account differently for the multiplicity of a point/ion during delocalization:

- unity, all points/ions get the same weight 1.
- atomic_decomposition, points get as much weight as they have atoms of a type in atomic_decomposition_rule,
- isotope_decomposition, points get as much weight as they have isotopes of a type in isotopic_decomposition_rule.

Any of these values:

- `unity`
- `atomic_decomposition`
- `isotopic_decomposition`

normalization: (required) *NX_CHAR*

How results of the kernel-density estimation were computed into quantities. By default the tool computes the total number (intensity) of ions or elements. Alternatively the tool can compute the total intensity, the composition, or the concentration of the ions/elements specified by the white list of elements in each voxel.

Any of these values:

- `total`
- `candidates`
- `composition`
- `concentration`

weight: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_ions]) {units=*NX_DIMENSIONLESS*} <=

Weighting factor, in atom probe, often termed multiplicity. The weighting factor is the multiplier with which the integrated intensity contribution from the point/ion gets multiplied. The delocalization computes the integrated intensity for each grid cell. Effectively, this is an explicitly evaluated kernel method where each specific position of an ion is replaced by a smoothing kernel. For atom probe weights are positive and integer specifically the multiplicity of the ion, in accordance with the respective rulesets as defined by weighting_model.

atomic_decomposition_rule: (optional) *NXmatch_filter*

A list of elements (via proton number) to consider for the atomic_decomposition weighting model. Elements must exist in the periodic table of elements and be specified by their number of protons. Values in match are isotope hash values using the following hashing rule $H = Z + 256*N$ with Z the number of protons and N the number of neutrons of the isotope. In the case of elements this hashing rule has the advantage that for elements the proton number is their hash value because N is zero. **method**: (required) *NX_CHAR* <=

Meaning of the filter: Whitelist specifies which entries with said value to include. Entries with all other values will be filtered out.

Blacklist specifies which entries with said value to exclude. Entries with all other values will be included.

Any of these values: `whitelist` | `blacklist`

match: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_atomic]) {units=*NX_UNITLESS*} <=

Array of values to filter according to method. For example, if the filter specifies [1, 5, 6] and method is whitelist, only entries with values matching 1, 5 or 6 will be processed. All other entries will be filtered out/not considered.

isotopic_decomposition_rule: (optional) *NXmatch_filter*

A list of isotopes (via proton and neutron number) to consider for the isotopic_decomposition weighting model. Isotopes must exist in the nuclid table. Values in match are isotope hash values using the following hashing rule $H = Z + 256*N$ with Z the number of protons and N the number of neutrons of the isotope. **method**: (required) *NX_CHAR* <=

Meaning of the filter: Whitelist specifies which entries with said value to include. Entries with all other values will be filtered out.

Blacklist specifies which entries with said value to exclude. Entries with all other values will be included.

Any of these values: `whitelist` | `blacklist`

match: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_isotopic]) {units=*NX_UNITLESS*} <=

Array of values to filter according to method. For example, if the filter specifies [1, 5, 6] and method is whitelist, only entries with values matching 1, 5 or 6 will be processed. All other entries will be filtered out/not considered.

grid: (required) *NXcg_grid*

The discretized domain/grid on which the delocalization is applied.

dimensionality: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

Any of these values: `1` | `2` | `3`

cardinality: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

The total number of cells/voxels of the grid.

origin: (required) *NX_NUMBER* (Rank: 1, Dimensions: [d]) <=

symmetry: (required) *NX_CHAR* <=

The symmetry of the lattice defining the shape of the unit cell.

Obligatory value: `cubic`

cell_dimensions: (required) *NX_NUMBER* (Rank: 1, Dimensions: [d]) {units=*NX_LENGTH*} <=

The unit cell dimensions according to the coordinate system defined under coordinate_system.

extent: (required) *NX_POSINT* (Rank: 1, Dimensions: [d]) {units=*NX_UNITLESS*} <=

Number of unit cells along each of the d unit vectors. The total number of cells, or grid points has to be the cardinality. If the grid has an irregular number of grid positions in each direction, as it could be for instance the case of a grid where all grid points outside some masking primitive are removed, this extent field should not be used. Instead use the coordinate field.

coordinate_system: (optional) *NX_CHAR*

Reference to or definition of a coordinate system with which the positions and directions are interpretable. If the coordinate system is not specified the paraprobe coordinate system is used.

identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

Integer which specifies the first index to be used for distinguishing identifiers for cells. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

kernel_size: (required) *NX_POSINT* (Rank: 1, Dimensions: [3]) {units=*NX_DIMENSIONLESS*}

Halfwidth of the kernel about the central voxel. The shape of the kernel is that of a cuboid of extent 2*kernel_extent[i] + 1 in each dimension i.

kernel_sigma: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

Sigma of the kernel in each dimension in the paraprobe coordinate_system with i = 0 is x, i = 1 is y, i = 2 is z.

kernel_mu: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

Expectation value of the kernel in each dimension in the paraprobe coordinate_system with i = 0 is x, i = 1 is y, i = 2 is z.

bounding_box: (required) *NXcg_hexahedron_set*

A tight axis-aligned bounding box about the grid.

is_axis_aligned: (required) *NX_BOOLEAN* {units=*NX_UNITLESS*} <=

For atom probe should be set to true.

identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

Integer which specifies the first index to be used for distinguishing hexahedra. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

boundaries: (optional) *NX_CHAR* (Rank: 1, Dimensions: [6])

Name of the boundaries. E.g. left, right, front, back, bottom, top, The field must have as many entries as there are number_of_boundaries.

boundary_conditions: (optional) *NX_INT* (Rank: 1, Dimensions: [6]) {units=*NX_UNITLESS*}

The boundary conditions for each boundary:

0 - undefined 1 - open 2 - periodic 3 - mirror 4 - von Neumann 5 - Dirichlet

hexahedron: (required) *NXcg_face_list_data_structure* <=

vertex_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

Integer which specifies the first index to be used for distinguishing identifiers for vertices. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

face_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

Integer which specifies the first index to be used for distinguishing identifiers for faces. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

vertices: (required) *NX_NUMBER* (Rank: 2, Dimensions: [8, 3]) {units=*NX_LENGTH*} <=

Positions of the vertices.

Users are encouraged to reduce the vertices to unique set of positions and vertices as this supports a more efficient storage of the geometry data. It is also possible though to store the vertex positions naively in which case vertices_are_unique is likely False. Naively here means that one for example stores each vertex of a triangle mesh even though many vertices are shared between triangles and thus the positions of these vertices do not have to be duplicated.

faces: (required) *NX_NUMBER* (Rank: 2, Dimensions: [6, 4]) {units=*NX_UNITLESS*}

Array of identifiers from vertices which describe each face.

The first entry is the identifier of the start vertex of the first face, followed by the second vertex of the first face, until the last vertex of the first face. Thereafter, the start vertex of the second face, the second vertex of the second face, and so on and so forth.

Therefore, summating over the number_of_vertices, allows to extract the vertex identifiers for the i-th face on the following index interval of the faces array: [$sum_i = 0}^{i = n-1}$, $sum_{i=0}^{i = n}$].

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [36]) {units=*NX_UNITLESS*}

Six equally formatted sextets chained together. For each sextett the first entry is an XDMF primitive topology key (here 5 for polygon), the second entry the XDMF primitive count value (here 4 because each face is a quad). The remaining four values are the vertex indices.

number_of_boundaries: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

How many distinct boundaries are distinguished? Most grids discretize a cubic or cuboidal region. In this case six sides can be distinguished, each making an own boundary.

scalar_field_magnitude: (required) *NXdata*

The result of the delocalization based on which subsequent iso-surfaces will be computed. In commercial software so far there is not a possibility to export such grid. **@long_name**: (optional) *NX_CHAR*

@signal: (required) *NX_CHAR <=*

@axes: (required) *NX_CHAR <=*

@xpos_indices: (required) *NX_CHAR*

@ypos_indices: (required) *NX_CHAR*

@zpos_indices: (required) *NX_CHAR*

intensity: (required) *NX_FLOAT* (Rank: 3, Dimensions: [n_z, n_y, n_x])

xpos: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_x])

Cell center of mass positions along x.

ypos: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_y])

Cell center of mass positions along y.

zpos: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_z])

Cell center of mass positions along z.

xdmf_intensity: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_xyz]) {units=*NX_ANY*}

Intensity of the field at given point

xdmf_xyz: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [n_xyz, 3]) {units=*NX_UNITLESS*}

Center of mass positions of each voxel for rendering the scalar field via XDMF in e.g. Paraview.

xdmf_topology: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [3*n_xyz]) {units=*NX_UNITLESS*}

XDMF topology for rendering in combination with xdmf_xyz the scalar field via XDFM in e.g. Paraview.

scalar_field_gradient: (required) *NXdata*

The three-dimensional gradient nabla operator applied to scalar_field_magnitude. **@signal**: (required) *NX_CHAR <=*

@axes: (required) *NX_CHAR <=*

@xpos_indices: (required) *NX_CHAR*

@ypos_indices: (required) *NX_CHAR*

@zpos_indices: (required) *NX_CHAR*

@long_name: (optional) *NX_CHAR*

intensity: (required) *NX_FLOAT* (Rank: 4, Dimensions: [n_z, n_y, n_x, 3])

xpos: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_x])

Cell center of mass positions along x.

ypos: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_y])

Cell center of mass positions along y.

zpos: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_z])

Cell center of mass positions along z.

xdmf_gradient: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [n_xyz, 3]) {units=*NX_ANY*}

The gradient vector.

xdmf_xyz: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [n_xyz, 3]) {units=*NX_UNITLESS*}

Center of mass positions of each voxel for rendering the scalar field via XDMF in e.g. Paraview.

xdmf_topology: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [3*n_xyz]) {units=*NX_UNITLESS*}

XDMF topology for rendering in combination with xdmf_xyz the scalar field via XDFM in e.g. Paraview.

iso_surface: (optional) *NXisocontour*

An iso-surface is the boundary between two regions across which the magnitude of a scalar field falls below/exceeds a threshold magnitude phi. For applications in atom probe microscopy the location and shape of such a boundary (set) is typically approximated by discretization. This yields a complex of not necessarily connected geometric primitives. Paraprobe-nanochem approximates this complex with a soup of triangles. **dimensionality**: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

isovalue: (required) *NX_NUMBER* {units=*NX_ANY*} *<=*

The threshold or iso-contour value.

marching_cubes: (required) *NXcg_marching_cubes*

Details about the specific marching cubes algorithm which was taken to compute the iso-surface. The grid is the delocalization grid. **implementation**: (required) *NX_CHAR* *<=*

Reference to the specific implementation of marching cubes used. The value placed here should be a DOI. If there are no specific DOI or details write not_further_specified, or give at least a free-text description. The program and version used is the specific paraprobe-nanochem.

triangle_soup: (optional) *NXcg_triangle_set*

The resulting triangle soup computed via marching cubes.

dimensionality: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

cardinality: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} *<=*

Integer which specifies the first index to be used for distinguishing triangles. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1].

triangles: (required) *NXcg_face_list_data_structure* <=

number_of_vertices: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

Number of vertices.

number_of_faces: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

Number of faces.

vertex_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

Integer which specifies the first index to be used for distinguishing identifiers for vertices. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1].

face_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

Integer which specifies the first index to be used for distinguishing identifiers for faces. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1].

vertices: (required) *NX_NUMBER* (Rank: 2, Dimensions: [i, 3]) {units=*NX_LENGTH*} <=

Positions of the vertices.

Users are encouraged to reduce the vertices to unique set of positions and vertices as this supports a more efficient storage of the geometry data. It is also possible though to store the vertex positions naively in which case vertices_are_unique is likely False. Naively here means that one for example stores each vertex of a triangle mesh even though many vertices are shared between triangles and thus the positions of these vertices do not have to be duplicated.

faces: (required) *NX_INT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*} <=

Array of identifiers from vertices which describe each face.

The first entry is the identifier of the start vertex of the first face, followed by the second vertex of the first face, until the last vertex of the first face. Thereafter, the start vertex of the second face, the second vertex of the second face, and so on and so forth.

Therefore, summating over the number_of_vertices, allows to extract the vertex identifiers for the i-th face on the following index interval of the faces array: [$sum_i = 0}^{i = n-1}$, $sum_{i=0}^{i = n}$].

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [n_f_tri_xdmf]) {units=*NX_UNITLESS*}

A list of as many tuples of XDMF topology key, XDMF number of vertices and a triple of vertex indices specifying each triangle. The total number of entries is n_f_tri * (1+1+3).

area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [j]) {units=*NX_AREA*}

edge_length: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [k, 3]) {units=*NX_LENGTH*}

Array of edge length values. For each triangle the edge length is reported for the edges traversed according to the sequence in which vertices are indexed in triangles.

interior_angle: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [j, 4]) {units=*NX_ANGLE*}

Array of interior angle values. For each triangle the angle is reported for the angle opposite to the edges which are traversed according to the sequence in which vertices are indexed in triangles.

center: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [j, 3]) {units=*NX_LENGTH*}

The center of mass of each triangle.

vertex_normal: (optional) *NXcg_unit_normal_set*

normals: (required) *NX_FLOAT* (Rank: 2, Dimensions: [j, 3]) {units=*NX_DIMENSIONLESS*} <=

Direction of each normal.

orientation: (optional) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

Qualifier how which specifically oriented normal to its primitive each normal represents.

- 0 - undefined
- 1 - outer
- 2 - inner

face_normal: (optional) *NXcg_unit_normal_set*

normals: (required) *NX_FLOAT* (Rank: 2, Dimensions: [k, 3]) {units=*NX_DIMENSIONLESS*} *<=*

Direction of each normal.

orientation: (optional) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

Qualifier how which specifically oriented normal to its primitive each normal represents.

- 0 - undefined
- 1 - outer
- 2 - inner

gradient_guide_magnitude: (required) *NX_FLOAT* (Rank: 1, Dimensions: [k]) {units=*NX_ANY*}

Triangle normals are oriented in the direction of the gradient vector of the local delocalized scalar field. $\sum_{x,y,z} \nabla {c_i}^2$.

gradient_guide_projection: (required) *NX_FLOAT* (Rank: 1, Dimensions: [k]) {units=*NX_ANY*}

Triangle normals are oriented in the direction of the gradient vector of the local delocalized scalar field. The projection variable here describes the cosine of the angle between the gradient direction and the normal direction vector. This is a descriptor of how parallel the projection is that is especially useful to document those triangles for whose projection is almost perpendicular.

volumetric_feature: (optional) *NXprocess*

Iso-surfaces of arbitrary scalar three-dimensional fields can show a complicated topology. Paraprobe-nanochem can run a DBScan-like clustering algorithm which performs a connectivity analysis on the triangle soup. This yields a set of connected features with their surfaces discretized by triangles. Currently, the tool distinguishes at most three types of features:

1. So-called objects, i.e. necessarily watertight features represented polyhedra.
2. So-called proxies, i.e. features that were replaced by a proxy mesh and made watertight.
3. Remaining triangle surface meshes of arbitrary shape and cardinality.

These features can be interpreted as microstructural features. Some of them may be precipitates, some of them may be poles, some of them may be segments of dislocation lines or other crystal defects which are decorated (or not) with solutes. **triangle_cluster_identifier**: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

The identifier which the triangle_soup connectivity analysis returned, which constitutes the first step of the volumetric_feature identification process.

feature_type_dict_keyword: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_feature_dict]) {units=*NX_UNITLESS*}

The array of keywords of feature_type dictionary.

feature_type_dict_value: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_feature_dict])

The array of values for each keyword of the feature_type dictionary.

feature_type: (required) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_ANY*}

The array of controlled keywords, need to be from feature_type_dict_keyword, which specify which type each feature triangle cluster belongs to. Keep in mind that not each feature is an object or proxy.

feature_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

The explicit identifier of features.

objects: (optional) *NXprocess*

Details for features which are (closed) objects. Identifier have to exist in feature_identifier. **feature_identifier**: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

volume: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_VOLUME*}

obb: (optional) *NXcg_hexahedron_set*

An oriented bounding box (OBB) to each object.

size: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [i, 3]) {units=*NX_LENGTH*}

Edge length of the oriented bounding box from largest to smallest value.

aspect: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [i, 2]) {units=*NX_DIMENSIONLESS*}

Oriented bounding box aspect ratio. YX versus ZY.

center: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [i, 3]) {units=*NX_LENGTH*} <=

Position of the geometric center, which often is but not necessarily has to be the center_of_mass of the hexahedrally-shaped sample/sample part.

hexahedra: (required) *NXcg_face_list_data_structure* <=

A simple approach to describe the entire set of hexahedra when the main intention is to store the shape of the hexahedra for visualization. **vertices**: (required) *NX_NUMBER* (Rank: 2, Dimensions: [k, 3]) {units=*NX_LENGTH*} <=

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

xdmf_feature_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

objects_close_to_edge: (optional) *NXprocess*

Details for all those objects close to edge, i.e. those which have at least one ion which lays closer to a modelled edge of the dataset than threshold. **feature_identifier**: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

volume: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_VOLUME*}

composition: (optional) *NXchemical_composition*

total: (required) *NX_NUMBER* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} <=

Total (count) relevant for normalization.

ION: (optional) *NXion* <=

charge: (required) *NX_INT*

isotope_vector: (required) *NX_UINT* <=

nuclid_list: (required) *NX_UINT* <=

count: (required) *NX_NUMBER* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} <=

Count or weight which, when divided by total, yields the composition of this element, isotope, molecule or ion.

objectID: (optional) *NXcg_polyhedron_set*

polyhedron: (required) *NXcg_face_list_data_structure* <=

vertices: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_v, 3]) {units=*NX_LENGTH*}

faces: (required) *NX_UINT* (Rank: 2, Dimensions: [n_f, 3]) {units=*NX_UNITLESS*}

face_normals: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_f, 3]) {units=*NX_LENGTH*}

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

xdmf_feature_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

ion_identifier: (optional) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Array of evaporation_identifier / ion_identifier which specify ions laying inside or on the surface of the feature.

objects_far_from_edge: (optional) *NXprocess*

Details for all those objects far from edge, i.e. those whose ions lay all at least threshold distant from a modelled edge of the dataset. **feature_identifier**: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

volume: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_VOLUME*}

composition: (optional) *NXchemical_composition*

total: (required) *NX_NUMBER* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} <=

Total (count) relevant for normalization.

ION: (optional) *NXion* <=

charge: (required) *NX_INT*

isotope_vector: (required) *NX_UINT* <=

nuclid_list: (required) *NX_UINT* <=

count: (required) *NX_NUMBER* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} <=

Count or weight which, when divided by total yields the composition of this element, isotope, molecule or ion.

objectID: (optional) *NXcg_polyhedron_set*

polyhedron: (required) *NXcg_face_list_data_structure* <=

vertices: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_v, 3]) {units=*NX_LENGTH*}

faces: (required) *NX_UINT* (Rank: 2, Dimensions: [n_f, 3]) {units=*NX_UNITLESS*}

face_normals: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_f, 3]) {units=*NX_LENGTH*}

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

xdmf_feature_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

ion_identifier: (optional) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Array of evaporation_identifier / ion_identifier which specify ions laying inside or on the surface of the feature.

proxies: (optional) *NXprocess*

Details for features which are so-called proxies, i.e. objects which have been reconstructed and combinatorially closed with processing their partial triangulated_surface_mesh (hole filling, refinement). Identifier have to exist in feature_identifier. **feature_identifier**: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

volume: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_VOLUME*}

proxies_close_to_edge: (optional) *NXprocess*

Details for those proxies close to edge, i.e. those which have at least one ion which lays closer to a modelled edge of the dataset than threshold. Identifier have to exist in feature_identifier. **feature_identifier**: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

volume: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_VOLUME*}

composition: (optional) *NXchemical_composition*

total: (required) *NX_NUMBER* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} *<=*

Total (count) relevant for normalization.

ION: (optional) *NXion* *<=*

count: (required) *NX_NUMBER* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} *<=*

Count or weight which, when divided by total yields the composition of this element, isotope, molecule or ion.

objectID: (optional) *NXcg_polyhedron_set*

polyhedron: (required) *NXcg_face_list_data_structure* *<=*

vertices: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_v, 3]) {units=*NX_LENGTH*}

faces: (required) *NX_UINT* (Rank: 2, Dimensions: [n_f, 3]) {units=*NX_UNITLESS*}

face_normals: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_f, 3]) {units=*NX_LENGTH*}

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

xdmf_feature_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

ion_identifier: (optional) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Array of evaporation_identifier / ion_identifier which specify ions laying inside or on the surface of the feature.

proxies_far_from_edge: (optional) *NXprocess*

Details for those proxies far from edge, i.e. those whose ions lay all at least threshold distant from a modelled edge of the dataset. **feature_identifier**: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

volume: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_VOLUME*}

composition: (optional) *NXchemical_composition*

total: (required) *NX_NUMBER* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} *<=*

Total (count) relevant for normalization.

ION: (optional) *NXion <=*

count: (required) *NX_NUMBER* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} *<=*

Count or weight which, when divided by total yields the composition of this element, isotope, molecule or ion.

objectID: (optional) *NXcg_polyhedron_set*

polyhedron: (required) *NXcg_face_list_data_structure <=*

vertices: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_v, 3]) {units=*NX_LENGTH*}

faces: (required) *NX_UINT* (Rank: 2, Dimensions: [n_f, 3]) {units=*NX_UNITLESS*}

face_normals: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_f, 3]) {units=*NX_LENGTH*}

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

xdmf_feature_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

ion_identifier: (optional) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Array of evaporation_identifier / ion_identifier which specify ions laying inside or on the surface of the feature.

interface_modelling: (optional) *NXprocess*

ion_multiplicity: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*}

The multiplicity whereby the ion position is accounted for irrespective whether the ion is considered as a decorator of the interface or not. As an example, with atom probe it is typically not possible to resolve the positions of the atoms which arrive at the detector as molecular ions. Therefore, an exemplar molecular ion of two carbon atoms can be considered to have a multiplicity of two to account that this molecular ion contributes two carbon atoms at the reconstructed location considering that the spatial resolution of atom probe experiments is limited.

decorator_multiplicity: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*}

The multiplicity whereby the ion position is accounted for when the ion is considered one which is a decorator of the interface.

initial_interface: (optional) *NXprocess*

The equation of the plane that is fitted initially.

point_normal_form: (required) *NX_FLOAT* (Rank: 1, Dimensions: [4]) {units=*NX_LENGTH*}

The four parameter $ax + by + cz + d = 0$ which define the plane.

MESH_CURR_PRE_DCOM_STEP: (optional) *NXcg_triangle_set*

The triangle surface mesh representing the interface model. Exported at some iteration before the next DCOM step. **dimensionality**: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

cardinality: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

area: (required) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*} <=

edge_length: (required) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_LENGTH*} <=

Array of edge length values. For each triangle the edge length is reported for the edges traversed according to the sequence in which vertices are indexed in triangles.

interior_angle: (required) *NX_NUMBER* (Rank: 2, Dimensions: [c, 4]) {units=*NX_ANGLE*} <=

Array of interior angle values. For each triangle the angle is reported for the angle opposite to the edges which are traversed according to the sequence in which vertices are indexed in triangles.

triangles: (required) *NXcg_face_list_data_structure* <=

dimensionality: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

number_of_vertices: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

number_of_faces: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

vertex_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

edge_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

face_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

face_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

vertices: (required) *NX_NUMBER* (Rank: 2, Dimensions: [i, 3]) {units=*NX_LENGTH*} <=

faces: (required) *NX_UINT* (Rank: 2, Dimensions: [j, 3]) {units=*NX_UNITLESS*}

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

face_normal: (required) *NXcg_unit_normal_set* <=

dimensionality: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

cardinality: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

normals: (required) *NX_FLOAT* (Rank: 2, Dimensions: [j, 3]) {units=*NX_LENGTH*} <=

Direction of each normal

orientation: (required) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

Qualifier how which specifically oriented normal to its primitive each normal represents.

- 0 - undefined
- 1 - outer
- 2 - inner

vertex_normal: (required) *NXcg_unit_normal_set* <=

dimensionality: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

cardinality: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

normals: (required) *NX_FLOAT* (Rank: 2, Dimensions: [j, 3]) {units=*NX_LENGTH*} *<=*

Direction of each normal

orientation: (required) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

Qualifier how which specifically oriented normal to its primitive each normal represents.

- 0 - undefined
- 1 - outer
- 2 - inner

MESH_CURR_POST_DCOM_STEP: (optional) *NXcg_triangle_set*

The triangle surface mesh representing the interface model. Exported at some iteration after the next DCOM step. **dimensionality**: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

cardinality: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} *<=*

area: (required) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*} *<=*

edge_length: (required) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_LENGTH*} *<=*

Array of edge length values. For each triangle the edge length is reported for the edges traversed according to the sequence in which vertices are indexed in triangles.

interior_angle: (required) *NX_NUMBER* (Rank: 2, Dimensions: [c, 4]) {units=*NX_ANGLE*} *<=*

Array of interior angle values. For each triangle the angle is reported for the angle opposite to the edges which are traversed according to the sequence in which vertices are indexed in triangles.

triangles: (required) *NXcg_face_list_data_structure* *<=*

dimensionality: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

number_of_vertices: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

number_of_faces: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

vertex_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} *<=*

face_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} *<=*

face_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

vertices: (required) *NX_NUMBER* (Rank: 2, Dimensions: [i, 3]) {units=*NX_LENGTH*} *<=*

faces: (required) *NX_UINT* (Rank: 2, Dimensions: [j, 3]) {units=*NX_UNITLESS*}

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

face_normal: (required) *NXcg_unit_normal_set <=*

dimensionality: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

cardinality: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

normals: (required) *NX_FLOAT* (Rank: 2, Dimensions: [j, 3]) {units=*NX_LENGTH*} *<=*

Direction of each normal

orientation: (required) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

Qualifier how which specifically oriented normal to its primitive each normal represents.

- 0 - undefined
- 1 - outer
- 2 - inner

vertex_normal: (required) *NXcg_unit_normal_set <=*

dimensionality: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

cardinality: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

normals: (required) *NX_FLOAT* (Rank: 2, Dimensions: [j, 3]) {units=*NX_LENGTH*} *<=*

Direction of each normal

orientation: (required) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

Qualifier how which specifically oriented normal to its primitive each normal represents.

- 0 - undefined
- 1 - outer
- 2 - inner

composition_analysis: (optional) *NXprocess*

xdmf_cylinder: (required) *NXcg_polyhedron_set*

The ROIs are defined as cylinders for the computations. To visualize these though we discretize them into regular n-gons. Using for instance a 360-gon, i.e. a regular n-gon with 360 edges resolves the lateral surface of each cylinder very finely so that they are rendered smoothly in visualization software. **dimensionality**: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

cardinality: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

center: (required) *NX_NUMBER* (Rank: 2, Dimensions: [i, 3]) {units=*NX_LENGTH*} <=

Position of the geometric center, which often is but not necessarily has to be the center_of_mass of the polyhedra.

roi_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing ROI cylinder. Identifiers are defined explicitly.

edge_contact: (optional) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

number_of_atoms: (optional) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

The number of atoms in each ROI.

number_of_ions: (optional) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

The number of ions in each ROI.

orientation: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [i, 3]) {units=*NX_LENGTH*}

The orientation of the ROI defined via a vector which points along the cylinder axis and whose length is the height of the cylinder.

polyhedra: (optional) *NXcg_face_list_data_structure* <=

ROI: (optional) *NXcollection*

signed_distance: (required) *NX_FLOAT*

In the direction of the ROI.

isotope: (required) *NX_UINT*

Hashvalue

performance: (recommended) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* <=

command_line_call: (optional) *NX_CHAR* <=

start_time: (recommended) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

total_elapsed_time: (required) *NX_NUMBER* <=

number_of_processes: (required) *NX_POSINT* <=

number_of_threads: (required) *NX_POSINT* <=

number_of_gpus: (required) *NX_POSINT* <=

CS_COMPUTER: (recommended) *NXcs_computer <=*

name: (recommended) *NX_CHAR <=*

operating_system: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

uuid: (optional) *NX_CHAR <=*

CS_CPU: (optional) *NXcs_cpu <=*

name: (optional) *NX_CHAR <=*

FABRICATION: (recommended) *NXfabrication <=*

identifier: (optional) *NX_CHAR <=*

capabilities: (optional) *NX_CHAR*

CS_GPU: (optional) *NXcs_gpu <=*

name: (optional) *NX_CHAR <=*

FABRICATION: (recommended) *NXfabrication <=*

identifier: (optional) *NX_CHAR <=*

capabilities: (optional) *NX_CHAR*

CS_MM_SYS: (optional) *NXcs_mm_sys <=*

total_physical_memory: (required) *NX_NUMBER <=*

CS_IO_SYS: (optional) *NXcs_io_sys <=*

CS_IO_OBJ: (required) *NXcs_io_obj <=*

technology: (required) *NX_CHAR <=*

max_physical_capacity: (required) *NX_NUMBER <=*

name: (optional) *NX_CHAR <=*

FABRICATION: (recommended) *NXfabrication <=*

identifier: (optional) *NX_CHAR <=*

capabilities: (optional) *NX_CHAR*

CS_PROFILING_EVENT: (required) *NXcs_profiling_event*

start_time: (optional) *NX_DATE_TIME <=*

end_time: (optional) *NX_DATE_TIME <=*

description: (required) *NX_CHAR <=*

elapsed_time: (required) *NX_NUMBER <=*

number_of_processes: (required) *NX_POSINT <=*

Specify if it was different from the number_of_processes in the NXcs_profiling super class.

number_of_threads: (required) *NX_POSINT <=*

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

number_of_gpus: (required) *NX_POSINT <=*

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

max_virtual_memory_snapshot: (recommended) *NX_NUMBER <=*

max_resident_memory_snapshot: (recommended) *NX_NUMBER <=*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_results_nanochem/ENTRY-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/analysis_description-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/config_filename-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/config_filename@version-attribute*
- */NXapm_paraprobe_results_nanochem/ENTRY/COORDINATE_SYSTEM_SET-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/definition-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/end_time-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/command_line_call-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_CPU-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_CPU/name-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_GPU-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_GPU/name-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_IO_SYS-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ-group*

- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/capabil field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identifie field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/max_physical_capacity-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/name-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/technology-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_MM_SYS-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_MM_SYS/total_physical_memory-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/description-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/elapsed_time-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/end_time-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_resident_memory_ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_gpus-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_processes-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_threads-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/start_time-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/name-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/operating_system-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/operating_system@version-attribute*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/uuid-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/end_time-field*

- */NXapm_paraprobe_results_nanochem/ENTRY/performance/number_of_gpus-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/number_of_processes-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/number_of_threads-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/start_time-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/performance/total_elapsed_time-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/composition_analysis-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/composition_analysis/xdmf_cylinder-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/composition_analysis/xdmf_cylinder/cardinality-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/composition_analysis/xdmf_cylinder/center-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/composition_analysis/xdmf_cylinder/dimensionality-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/composition_analysis/xdmf_cylinder/edge_contact-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/composition_analysis/xdmf_cylinder/number_of_atoms-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/composition_analysis/xdmf_cylinder/number_of_ions-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/composition_analysis/xdmf_cylinder/orientation-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/composition_analysis/xdmf_cylinder/polyhedra-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/composition_analysis/xdmf_cylinder/ROI-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/composition_analysis/xdmf_cylinder/ROI/isotope-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/composition_analysis/xdmf_cylinder/ROI/signed_distance-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/composition_analysis/xdmf_cylinder/roi_identifier-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/decorator_multiplicity-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/initial_interface-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/initial_interface/point_normal_form-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/ion_multiplicity-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/area-field*

- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/cardinality-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/dimensiona field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/edge_length field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/face_norma group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/face_norma field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/face_norma field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/face_norma field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/face_norma field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/identifier_o field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/interior_an field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/triangles-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/triangles/di field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/triangles/fa field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/triangles/fa field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/triangles/fa field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/triangles/nu field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/triangles/nu field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/triangles/ve field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/triangles/ve field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/triangles/xa field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/vertex_norm group*

- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/vertex_norm field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/vertex_norm field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/vertex_norm field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/vertex_norm field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/area-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/cardinality-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/dimensionali field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/edge_length-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/face_normal group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/face_normal field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/face_normal field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/face_normal field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/face_normal field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/identifier_off field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/interior_ang field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/triangles-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/triangles/dim field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/triangles/edg field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/triangles/face field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/triangles/face field*

- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/triangles/face field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/triangles/num field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/triangles/num field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/triangles/ver field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/triangles/ver field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/triangles/xdm field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/vertex_norm group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/vertex_norm field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/vertex_norm field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/vertex_norm field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/vertex_norm field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/atomic_decomposition_rule group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/atomic_decomposition_rule field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/atomic_decomposition_rule field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/bounda field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/bounda field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/hexahed group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/hexahed field*

- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/hexahe field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/hexahe field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/hexahe field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/hexahe field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/hexahe field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/identifi field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/is_axis_ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/cardinality-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/cell_dimensions-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/coordinate_system-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/dimensionality-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/extent-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/identifier_offset-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/kernel_mu-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/kernel_sigma-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/kernel_size-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/origin-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_gradient-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_gradient/ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_gradient/ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_gradient/ field*

- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_gradient/ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_gradient/ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_gradient/ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_gradient/ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_gradient@ attribute*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_gradient@ attribute*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_gradient@ attribute*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_gradient@ attribute*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_gradient@ attribute*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_gradient@ attribute*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitud group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitud field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitud field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitud field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitud field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitud field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitud field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitud field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitud attribute*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitud attribute*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitud attribute*

- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitud attribute*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitud attribute*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitud attribute*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/symmetry-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/dimensionality-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/isovalue-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/marching_cube group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/marching_cube field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/c field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/c field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/i field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field*

- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/t field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/v group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/v field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/v field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/v field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/v field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/v group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/v field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/v group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/v field

- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\
field

- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*

- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group*

- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/iso_surface/triangle_soup/\ field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/isotopic_decomposition_ru group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/isotopic_decomposition_ru field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/isotopic_decomposition_ru field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/normalization-field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/weight-field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/weighting_model-field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/window-group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/window/bitdepth-field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/window/mask-field

- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/window/number_of_ions-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/program-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/program@version-attribute*
- */NXapm_paraprobe_results_nanochem/ENTRY/results_path-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/start_time-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/status-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/USER-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/USER/address-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/USER/affiliation-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/USER/email-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/USER/name-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/USER/orcid-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/USER/orcid_platform-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/USER/role-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/USER/social_media_name-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/USER/social_media_platform-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/USER/telephone_number-field*
- */NXapm_paraprobe_results_nanochem/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_results_nanochem.nxdl.xml

NXapm_paraprobe_results_ranger

Status:

application definition, extends *NXobject*

Description:

Results of a paraprobe-ranger tool run.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ions: The total number of ions in the reconstruction.

n_ivec_max: Maximum number of allowed atoms per (molecular) ion (fragment). Needs to match maximum_number_of_atoms_per_molecular_ion.

Groups cited:
NXcoordinate_system_set, *NXcs_computer*, *NXcs_cpu*, *NXcs_filter_boolean_mask*, *NXcs_gpu*, *NXcs_io_obj*, *NXcs_io_sys*, *NXcs_mm_sys*, *NXcs_profiling_event*, *NXcs_profiling*, *NXentry*, *NXfabrication*, *NXion*, *NXprocess*, *NXtransformations*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_results_ranger`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

start_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file was started, i.e. the paraprobe-tool executable started as a process.

end_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file were completed and the paraprobe-tool executable exited as a process.

config_filename: (required) *NX_CHAR*

The path and name of the config file for this analysis.

@version: (required) *NX_CHAR*

At least SHA256 strong hash of the specific config_file for tracking provenance.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

status: (required) *NX_CHAR*

A statement whether the paraprobe-tool executable managed to process the analysis or failed prematurely.

This status is written to the results file after the end_time at which point the executable must not compute any analysis. Only when this status message is present and shows *success*, the user should consider the results. In all other cases it might be that the executable has terminated prematurely or another error occurred.

Any of these values: `success` | `failure`

USER: (recommended) *NXuser <=*

If used, contact information and eventually details of at least the person who performed this analysis. **name**: (required) *NX_CHAR <=*

affiliation: (recommended) *NX_CHAR <=*

address: (optional) *NX_CHAR <=*

email: (recommended) *NX_CHAR <=*

orcid: (recommended) *NX_CHAR <=*

orcid_platform: (recommended) *NX_CHAR <=*

telephone_number: (optional) *NX_CHAR <=*

role: (recommended) *NX_CHAR <=*

social_media_name: (optional) *NX_CHAR <=*

social_media_platform: (optional) *NX_CHAR <=*

COORDINATE_SYSTEM_SET: (optional) *NXcoordinate_system_set*

Details about the coordinate system conventions used.

TRANSFORMATIONS: (required) *NXtransformations <=*

The individual coordinate systems which should be used. Field names should be prefixed with the following controlled terms indicating which individual coordinate system is described:

- paraprobe
- lab
- specimen
- laser
- leap
- detector
- recon

PROCESS: (optional) *NXprocess <=*

apply_existent_ranging: (optional) *NXprocess*

Paraprobe-ranger loads the iontypes and evaluates for each ion on which iontype it matches. If it matches on none, the ion is considered of the default unknown type with a 0 as its respective value in the iontypes array. **maximum_number_of_atoms_per_molecular_ion**: (required) *NX_POSINT* {units=*NX_UNITLESS*}

The length of the isotope_vector used to describe molecular ions.

iontypes: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*}

The iontype ID for each ion that was best matching, stored in the order of the evaporation sequence ID. The here computed iontypes do not take into account the charge state of the ion which is equivalent to interpreting a RNG and RRNG range files for each ion in such a way that only the elements of which a (molecular) ion is build are considered. By contrast, charged_iontypes takes into account also the charge state.

ION: (required) *NXion*

isotope_vector: (required) *NX_UINT* <=

nuclid_list: (recommended) *NX_UINT* <=

charge_state: (required) *NX_INT* <=

mass_to_charge_range: (required) *NX_FLOAT* <=

window: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies exactly all those ions ranged irrespective the type they ended up with. **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of ions covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*} <=

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

molecular_ion_search: (optional) *NXprocess*

Paraprobe-ranger performs a combinatorial search over all possible or a reduced set of nuclids to identify into which ions these can be composed. **isotope_vector_matrix**: (required) *NX_UINT* (Rank: 2, Dimensions: [i, 32]) {units=*NX_UNITLESS*}

The main result is the list of molecular ions, here formatted according to the definitions of a set of isotope_vectors as detailed in *NXion*.

mass_to_charge_state_ratio: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_ANY*}

The mass-to-charge-state ratio of each molecular ion without considering relativistic or quantum effects.

mass: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_ANY*}

The mass of each molecular ion without considering relativistic or quantum effects.

charge_state: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_CHARGE*}

The charge_state of each molecular ion.

natural_abundance_product: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_DIMENSIONLESS*}

The product of the natural abundance of the isotopes building each molecular ion. Further details are available in *NXapm_paraprobe_config_ranger*.

composition_product: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_DIMENSIONLESS*}

The product of the natural abundance of the isotopes building each molecular ion. Further details are available in *NXapm_paraprobe_config_ranger*.

number_of_disjoint_nuclids: (optional) *NX_POSINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

The number of disjoint nuclids for each molecular ion.

number_of_nuclids: (optional) *NX_POSINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

The number of nuclids for each molecular ion.

check_existent_ranging: (optional) *NXprocess*

Paraprobe-ranger loads iontypes and evaluates for each ion on which iontype it matches. If it matches on none, the ion is considered of the default unknown type with a 0 as its respective value in the iontypes array. In contrast to use_existent_ranging this process does neither needs measured ion position nor mass-to-charge-state ratio values. **maximum_number_of_atoms_per_molecular_ion**: (required) *NX_POSINT* {units=*NX_UNITLESS*}

The length of the isotope_vector used to describe molecular ions.

charged_ION: (required) *NXion*

isotope_vector: (required) *NX_UINT* <=

nuclid_list: (recommended) *NX_UINT* <=

charge_state: (required) *NX_INT* <=

mass_to_charge_range: (required) *NX_FLOAT* <=

performance: (required) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* <=

command_line_call: (optional) *NX_CHAR* <=

start_time: (recommended) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

total_elapsed_time: (required) *NX_NUMBER* <=

number_of_processes: (required) *NX_POSINT* <=

number_of_threads: (required) *NX_POSINT* <=

number_of_gpus: (required) *NX_POSINT* <=

CS_COMPUTER: (recommended) *NXcs_computer* <=

- **name**: (recommended) *NX_CHAR* <=
- **operating_system**: (required) *NX_CHAR* <=
  - **@version**: (required) *NX_CHAR* <=
- **uuid**: (optional) *NX_CHAR* <=
- **CS_CPU**: (optional) *NXcs_cpu* <=
  - **name**: (optional) *NX_CHAR* <=
  - **FABRICATION**: (recommended) *NXfabrication* <=
    - **identifier**: (optional) *NX_CHAR* <=
    - **capabilities**: (optional) *NX_CHAR*
- **CS_GPU**: (optional) *NXcs_gpu* <=
  - **name**: (optional) *NX_CHAR* <=
  - **FABRICATION**: (recommended) *NXfabrication* <=
    - **identifier**: (optional) *NX_CHAR* <=
    - **capabilities**: (optional) *NX_CHAR*
- **CS_MM_SYS**: (optional) *NXcs_mm_sys* <=
  - **total_physical_memory**: (required) *NX_NUMBER* <=
- **CS_IO_SYS**: (optional) *NXcs_io_sys* <=
  - **CS_IO_OBJ**: (required) *NXcs_io_obj* <=
    - **technology**: (required) *NX_CHAR* <=
    - **max_physical_capacity**: (required) *NX_NUMBER* <=
    - **name**: (optional) *NX_CHAR* <=
    - **FABRICATION**: (recommended) *NXfabrication* <=
      - **identifier**: (optional) *NX_CHAR* <=
      - **capabilities**: (optional) *NX_CHAR*
- **CS_PROFILING_EVENT**: (required) *NXcs_profiling_event*
  - **start_time**: (optional) *NX_DATE_TIME* <=
  - **end_time**: (optional) *NX_DATE_TIME* <=
  - **description**: (required) *NX_CHAR* <=
  - **elapsed_time**: (required) *NX_NUMBER* <=
  - **number_of_processes**: (required) *NX_POSINT* <=

    Specify if it was different from the number_of_processes in the NXcs_profiling super class.
  - **number_of_threads**: (required) *NX_POSINT* <=

    Specify if it was different from the number_of_threads in the NXcs_profiling super class.
  - **number_of_gpus**: (required) *NX_POSINT* <=

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

max_virtual_memory_snapshot: (recommended) *NX_NUMBER* *<=*

max_resident_memory_snapshot: (recommended) *NX_NUMBER* *<=*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_results_ranger/ENTRY-group*
- */NXapm_paraprobe_results_ranger/ENTRY/analysis_description-field*
- */NXapm_paraprobe_results_ranger/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_results_ranger/ENTRY/config_filename-field*
- */NXapm_paraprobe_results_ranger/ENTRY/config_filename@version-attribute*
- */NXapm_paraprobe_results_ranger/ENTRY/COORDINATE_SYSTEM_SET-group*
- */NXapm_paraprobe_results_ranger/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-group*
- */NXapm_paraprobe_results_ranger/ENTRY/definition-field*
- */NXapm_paraprobe_results_ranger/ENTRY/end_time-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance-group*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/command_line_call-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER-group*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_CPU-group*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION-group*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_CPU/name-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_GPU-group*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION-group*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_GPU/name-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_IO_SYS-group*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ-group*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION-group*

- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/max_physical_capacity-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/name-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/technology-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_MM_SYS-group*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_MM_SYS/total_physical_memory-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT-group*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/description-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/elapsed_time-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/end_time-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_resident_memory_snapshot-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_snapshot-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_gpus-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_processes-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_threads-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/start_time-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/name-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/operating_system-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/operating_system@version-attribute*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/uuid-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/end_time-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/number_of_gpus-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/number_of_processes-field*

- */NXapm_paraprobe_results_ranger/ENTRY/performance/number_of_threads-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/start_time-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/total_elapsed_time-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS-group*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/apply_existent_ranging-group*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/apply_existent_ranging/ION-group*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/apply_existent_ranging/ION/charge_state-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/apply_existent_ranging/ION/isotope_vector-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/apply_existent_ranging/ION/mass_to_charge_range-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/apply_existent_ranging/ION/nuclid_list-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/apply_existent_ranging/iontypes-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/apply_existent_ranging/maximum_number_of_atoms_per_molecular_ion field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/apply_existent_ranging/window-group*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/apply_existent_ranging/window/bitdepth-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/apply_existent_ranging/window/mask-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/apply_existent_ranging/window/number_of_ions-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/check_existent_ranging-group*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/check_existent_ranging/charged_ION-group*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/check_existent_ranging/charged_ION/charge_state-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/check_existent_ranging/charged_ION/isotope_vector-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/check_existent_ranging/charged_ION/mass_to_charge_range-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/check_existent_ranging/charged_ION/nuclid_list-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/check_existent_ranging/maximum_number_of_atoms_per_molecular_ion field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/molecular_ion_search-group*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/molecular_ion_search/charge_state-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/molecular_ion_search/composition_product-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/molecular_ion_search/isotope_vector_matrix-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/molecular_ion_search/mass-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/molecular_ion_search/mass_to_charge_state_ratio-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/molecular_ion_search/natural_abundance_product-field*

- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/molecular_ion_search/number_of_disjoint_nuclids-field*
- */NXapm_paraprobe_results_ranger/ENTRY/PROCESS/molecular_ion_search/number_of_nuclids-field*
- */NXapm_paraprobe_results_ranger/ENTRY/program-field*
- */NXapm_paraprobe_results_ranger/ENTRY/program@version-attribute*
- */NXapm_paraprobe_results_ranger/ENTRY/results_path-field*
- */NXapm_paraprobe_results_ranger/ENTRY/start_time-field*
- */NXapm_paraprobe_results_ranger/ENTRY/status-field*
- */NXapm_paraprobe_results_ranger/ENTRY/USER-group*
- */NXapm_paraprobe_results_ranger/ENTRY/USER/address-field*
- */NXapm_paraprobe_results_ranger/ENTRY/USER/affiliation-field*
- */NXapm_paraprobe_results_ranger/ENTRY/USER/email-field*
- */NXapm_paraprobe_results_ranger/ENTRY/USER/name-field*
- */NXapm_paraprobe_results_ranger/ENTRY/USER/orcid-field*
- */NXapm_paraprobe_results_ranger/ENTRY/USER/orcid_platform-field*
- */NXapm_paraprobe_results_ranger/ENTRY/USER/role-field*
- */NXapm_paraprobe_results_ranger/ENTRY/USER/social_media_name-field*
- */NXapm_paraprobe_results_ranger/ENTRY/USER/social_media_platform-field*
- */NXapm_paraprobe_results_ranger/ENTRY/USER/telephone_number-field*
- */NXapm_paraprobe_results_ranger/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_results_ranger.nxdl.xml

NXapm_paraprobe_results_selector

Status:

application definition, extends *NXobject*

Description:

Results of a paraprobe-selector tool run.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ions: The total number of ions in the reconstruction.

Groups cited:
NXcoordinate_system_set, *NXcs_computer*, *NXcs_cpu*, *NXcs_filter_boolean_mask*, *NXcs_gpu*, *NXcs_io_obj*, *NXcs_io_sys*, *NXcs_mm_sys*, *NXcs_profiling_event*, *NXcs_profiling*, *NXentry*, *NXfabrication*, *NXprocess*, *NXtransformations*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_results_selector`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

start_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file was started, i.e. the paraprobe-tool executable started as a process.

end_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file were completed and the paraprobe-tool executable exited as a process.

config_filename: (required) *NX_CHAR*

The absolute path and name of the config file for this analysis.

@version: (required) *NX_CHAR*

At least SHA256 strong hash of the specific config_file for tracking provenance.

status: (required) *NX_CHAR*

A statement whether the paraprobe-tool executable managed to process the analysis or failed prematurely.

This status is written to the results file after the end_time at which point the executable must not compute any analysis. Only when this status message is present and shows *success*, the user should consider the results. In all other cases it might be that the executable has terminated prematurely or another error occurred.

Any of these values: `success` | `failure`

USER: (recommended) *NXuser <=*

If used, contact information and eventually details of at least the person who performed this analysis. **name**: (required) *NX_CHAR <=*

affiliation: (recommended) *NX_CHAR* *<=*

address: (optional) *NX_CHAR* *<=*

email: (recommended) *NX_CHAR* *<=*

orcid: (recommended) *NX_CHAR* *<=*

orcid_platform: (recommended) *NX_CHAR* *<=*

telephone_number: (optional) *NX_CHAR* *<=*

role: (recommended) *NX_CHAR* *<=*

social_media_name: (optional) *NX_CHAR* *<=*

social_media_platform: (optional) *NX_CHAR* *<=*

COORDINATE_SYSTEM_SET: (optional) *NXcoordinate_system_set*

Details about the coordinate system conventions used.

TRANSFORMATIONS: (required) *NXtransformations* *<=*

The individual coordinate systems which should be used. Field names should be prefixed with the following controlled terms indicating which individual coordinate system is described:

- paraprobe
- lab
- specimen
- laser
- leap
- detector
- recon

PROCESS: (optional) *NXprocess* *<=*

window: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies which of the ions in the dataset were selected to become included in the region-of-interest (ROI). **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of ions covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} *<=*

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*} *<=*

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe.

The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

- **performance**: (recommended) *NXcs_profiling*
  - **command_line_call**: (optional) *NX_CHAR* <=
  - **start_time**: (recommended) *NX_DATE_TIME* <=
  - **end_time**: (recommended) *NX_DATE_TIME* <=
  - **total_elapsed_time**: (required) *NX_NUMBER* <=
  - **number_of_processes**: (required) *NX_POSINT* <=
  - **number_of_threads**: (required) *NX_POSINT* <=
  - **number_of_gpus**: (required) *NX_POSINT* <=
  - **CS_COMPUTER**: (recommended) *NXcs_computer* <=
    - **name**: (recommended) *NX_CHAR* <=
    - **operating_system**: (required) *NX_CHAR* <=
      - **@version**: (required) *NX_CHAR* <=
    - **uuid**: (optional) *NX_CHAR* <=
    - **CS_CPU**: (optional) *NXcs_cpu* <=
      - **name**: (optional) *NX_CHAR* <=
      - **FABRICATION**: (recommended) *NXfabrication* <=
        - **identifier**: (optional) *NX_CHAR* <=
        - **capabilities**: (optional) *NX_CHAR*
    - **CS_GPU**: (optional) *NXcs_gpu* <=
      - **name**: (optional) *NX_CHAR* <=
      - **FABRICATION**: (recommended) *NXfabrication* <=
        - **identifier**: (optional) *NX_CHAR* <=
        - **capabilities**: (optional) *NX_CHAR*
    - **CS_MM_SYS**: (optional) *NXcs_mm_sys* <=
      - **total_physical_memory**: (required) *NX_NUMBER* <=
    - **CS_IO_SYS**: (optional) *NXcs_io_sys* <=
      - **CS_IO_OBJ**: (required) *NXcs_io_obj* <=
        - **technology**: (required) *NX_CHAR* <=
        - **max_physical_capacity**: (required) *NX_NUMBER* <=
        - **name**: (optional) *NX_CHAR* <=
        - **FABRICATION**: (recommended) *NXfabrication* <=
          - **identifier**: (optional) *NX_CHAR* <=
          - **capabilities**: (optional) *NX_CHAR*
    - **CS_PROFILING_EVENT**: (required) *NXcs_profiling_event*

start_time: (optional) *NX_DATE_TIME* <=

end_time: (optional) *NX_DATE_TIME* <=

description: (required) *NX_CHAR* <=

elapsed_time: (required) *NX_NUMBER* <=

number_of_processes: (required) *NX_POSINT* <=

Specify if it was different from the number_of_processes in the NXcs_profiling super class.

number_of_threads: (required) *NX_POSINT* <=

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

number_of_gpus: (required) *NX_POSINT* <=

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

max_virtual_memory_snapshot: (recommended) *NX_NUMBER* <=

max_resident_memory_snapshot: (recommended) *NX_NUMBER* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_results_selector/ENTRY-group*
- */NXapm_paraprobe_results_selector/ENTRY/analysis_description-field*
- */NXapm_paraprobe_results_selector/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_results_selector/ENTRY/config_filename-field*
- */NXapm_paraprobe_results_selector/ENTRY/config_filename@version-attribute*
- */NXapm_paraprobe_results_selector/ENTRY/COORDINATE_SYSTEM_SET-group*
- */NXapm_paraprobe_results_selector/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-group*
- */NXapm_paraprobe_results_selector/ENTRY/definition-field*
- */NXapm_paraprobe_results_selector/ENTRY/end_time-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance-group*
- */NXapm_paraprobe_results_selector/ENTRY/performance/command_line_call-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER-group*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_CPU-group*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION-group*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_CPU/name-field*

- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_GPU-group*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION-group*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_GPU/name-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS-group*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ-group*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION-group*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/capabiliti field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/max_physical_capacity-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/name-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/technology-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_MM_SYS-group*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_MM_SYS/total_physical_memory-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT-group*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/description-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/elapsed_time-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/end_time-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_resident_memory_s field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_sn field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_gpus-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_processes-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_threads-field*

- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/start_time-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/name-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/operating_system-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/operating_system@version-attribute*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/uuid-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/end_time-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/number_of_gpus-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/number_of_processes-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/number_of_threads-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/start_time-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/total_elapsed_time-field*
- */NXapm_paraprobe_results_selector/ENTRY/PROCESS-group*
- */NXapm_paraprobe_results_selector/ENTRY/PROCESS/window-group*
- */NXapm_paraprobe_results_selector/ENTRY/PROCESS/window/bitdepth-field*
- */NXapm_paraprobe_results_selector/ENTRY/PROCESS/window/mask-field*
- */NXapm_paraprobe_results_selector/ENTRY/PROCESS/window/number_of_ions-field*
- */NXapm_paraprobe_results_selector/ENTRY/program-field*
- */NXapm_paraprobe_results_selector/ENTRY/program@version-attribute*
- */NXapm_paraprobe_results_selector/ENTRY/start_time-field*
- */NXapm_paraprobe_results_selector/ENTRY/status-field*
- */NXapm_paraprobe_results_selector/ENTRY/USER-group*
- */NXapm_paraprobe_results_selector/ENTRY/USER/address-field*
- */NXapm_paraprobe_results_selector/ENTRY/USER/affiliation-field*
- */NXapm_paraprobe_results_selector/ENTRY/USER/email-field*
- */NXapm_paraprobe_results_selector/ENTRY/USER/name-field*
- */NXapm_paraprobe_results_selector/ENTRY/USER/orcid-field*
- */NXapm_paraprobe_results_selector/ENTRY/USER/orcid_platform-field*
- */NXapm_paraprobe_results_selector/ENTRY/USER/role-field*
- */NXapm_paraprobe_results_selector/ENTRY/USER/social_media_name-field*
- */NXapm_paraprobe_results_selector/ENTRY/USER/social_media_platform-field*
- */NXapm_paraprobe_results_selector/ENTRY/USER/telephone_number-field*
- */NXapm_paraprobe_results_selector/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_results_selector.nxdl.xml

NXapm_paraprobe_results_spatstat

Status:

application definition, extends *NXobject*

Description:

Results of a paraprobe-spatstat tool run.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ions: The total number of ions in the reconstruction.

Groups cited:
NXcoordinate_system_set, *NXcs_computer*, *NXcs_cpu*, *NXcs_filter_boolean_mask*, *NXcs_gpu*, *NXcs_io_obj*, *NXcs_io_sys*, *NXcs_mm_sys*, *NXcs_profiling_event*, *NXcs_profiling*, *NXentry*, *NXfabrication*, *NXprocess*, *NXtransformations*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_results_spatstat`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

start_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file was started, i.e. the paraprobe-tool executable started as a process.

end_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file were completed and the paraprobe-tool executable exited as a process.

config_filename: (required) *NX_CHAR*

The absolute path and name of the config file for this analysis.

@version: (required) *NX_CHAR*

At least SHA256 strong hash of the specific config_file for tracking provenance.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

status: (required) *NX_CHAR*

A statement whether the paraprobe-tool executable managed to process the analysis or failed prematurely.

This status is written to the results file after the end_time at which point the executable must not compute any analysis. Only when this status message is present and shows *success*, the user should consider the results. In all other cases it might be that the executable has terminated prematurely or another error occurred.

Any of these values: `success` | `failure`

USER: (recommended) *NXuser* <=

If used, contact information and eventually details of at least the person who performed this analysis. **name**: (required) *NX_CHAR* <=

affiliation: (recommended) *NX_CHAR* <=

address: (optional) *NX_CHAR* <=

email: (recommended) *NX_CHAR* <=

orcid: (recommended) *NX_CHAR* <=

orcid_platform: (recommended) *NX_CHAR* <=

telephone_number: (optional) *NX_CHAR* <=

role: (recommended) *NX_CHAR* <=

social_media_name: (optional) *NX_CHAR* <=

social_media_platform: (optional) *NX_CHAR* <=

COORDINATE_SYSTEM_SET: (optional) *NXcoordinate_system_set*

Details about the coordinate system conventions used.

TRANSFORMATIONS: (required) *NXtransformations* <=

The individual coordinate systems which should be used. Field names should be prefixed with the following controlled terms indicating which individual coordinate system is described:

- paraprobe
- lab
- specimen
- laser
- leap
- detector

- recon

PROCESS: (optional) *NXprocess* <=

iontypes_randomized: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*}

The iontype ID for each ion that was assigned to each ion during the randomization of the ionlabels. Iontype labels are just permuted but the total number of values for each iontype stay the same.

The order matches the iontypes array from a given ranging results as is specified in the configuration settings inside the specific config_filename that was used for this spatstat analysis.

window: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies which of the ions in the dataset were analyzed.

number_of_ions: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of ions covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*} <=

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

knn: (optional) *NXprocess*

K-nearest neighbor statistics.

distance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_LENGTH*}

Right boundary of the binning.

probability_mass: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_DIMENSIONLESS*}

cumulated: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Cumulated

cumulated_normalized: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_DIMENSIONLESS*}

Cumulated and normalized by total counts

rdf: (optional) *NXprocess*

Radial distribution statistics.

distance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_LENGTH*}

Right boundary of the binning.

probability_mass: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_DIMENSIONLESS*}

cumulated: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Cumulated

cumulated_normalized: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_DIMENSIONLESS*}

Cumulated and normalized by total counts

performance: (recommended) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* <=

command_line_call: (optional) *NX_CHAR* <=

start_time: (recommended) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

total_elapsed_time: (required) *NX_NUMBER* <=

number_of_processes: (required) *NX_POSINT* <=

number_of_threads: (required) *NX_POSINT* <=

number_of_gpus: (required) *NX_POSINT* <=

CS_COMPUTER: (recommended) *NXcs_computer* <=

name: (recommended) *NX_CHAR* <=

operating_system: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

uuid: (optional) *NX_CHAR* <=

CS_CPU: (optional) *NXcs_cpu* <=

name: (optional) *NX_CHAR* <=

FABRICATION: (recommended) *NXfabrication* <=

identifier: (optional) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CS_GPU: (optional) *NXcs_gpu* <=

name: (optional) *NX_CHAR* <=

FABRICATION: (recommended) *NXfabrication* <=

identifier: (optional) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CS_MM_SYS: (optional) *NXcs_mm_sys* <=

- **total_physical_memory**: (required) *NX_NUMBER <=*
- **CS_IO_SYS**: (optional) *NXcs_io_sys <=*
  - **CS_IO_OBJ**: (required) *NXcs_io_obj <=*
    - **technology**: (required) *NX_CHAR <=*
    - **max_physical_capacity**: (required) *NX_NUMBER <=*
    - **name**: (optional) *NX_CHAR <=*
    - **FABRICATION**: (recommended) *NXfabrication <=*
      - **identifier**: (optional) *NX_CHAR <=*
      - **capabilities**: (optional) *NX_CHAR*
- **CS_PROFILING_EVENT**: (required) *NXcs_profiling_event*
  - **start_time**: (optional) *NX_DATE_TIME <=*
  - **end_time**: (optional) *NX_DATE_TIME <=*
  - **description**: (required) *NX_CHAR <=*
  - **elapsed_time**: (required) *NX_NUMBER <=*
  - **number_of_processes**: (required) *NX_POSINT <=*

    Specify if it was different from the number_of_processes in the NXcs_profiling super class.
  - **number_of_threads**: (required) *NX_POSINT <=*

    Specify if it was different from the number_of_threads in the NXcs_profiling super class.
  - **number_of_gpus**: (required) *NX_POSINT <=*

    Specify if it was different from the number_of_threads in the NXcs_profiling super class.
  - **max_virtual_memory_snapshot**: (recommended) *NX_NUMBER <=*
  - **max_resident_memory_snapshot**: (recommended) *NX_NUMBER <=*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_results_spatstat/ENTRY-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/analysis_description-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/config_filename-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/config_filename@version-attribute*
- */NXapm_paraprobe_results_spatstat/ENTRY/COORDINATE_SYSTEM_SET-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/definition-field*

- */NXapm_paraprobe_results_spatstat/ENTRY/end_time-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/command_line_call-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_CPU-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_CPU/name-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_GPU-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_GPU/name-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_IO_SYS-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/capabiliti field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/max_physical_capacity-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/name-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/technology-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_MM_SYS-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_MM_SYS/total_physical_memory-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/description-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/elapsed_time-field*

- /NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/end_time-field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_resident_memory_sr field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_sna field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_gpus-field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_processes-field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_threads-field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/start_time-field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/name-field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/operating_system-field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/operating_system@version-attribute
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/uuid-field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/current_working_directory-field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/end_time-field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/number_of_gpus-field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/number_of_processes-field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/number_of_threads-field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/start_time-field
- /NXapm_paraprobe_results_spatstat/ENTRY/performance/total_elapsed_time-field
- /NXapm_paraprobe_results_spatstat/ENTRY/PROCESS-group
- /NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/iontypes_randomized-field
- /NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/knn-group
- /NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/knn/cumulated-field
- /NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/knn/cumulated_normalized-field
- /NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/knn/distance-field
- /NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/knn/probability_mass-field
- /NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/rdf-group
- /NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/rdf/cumulated-field
- /NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/rdf/cumulated_normalized-field
- /NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/rdf/distance-field
- /NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/rdf/probability_mass-field
- /NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/window-group

- */NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/window/bitdepth-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/window/mask-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/PROCESS/window/number_of_ions-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/program-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/program@version-attribute*
- */NXapm_paraprobe_results_spatstat/ENTRY/results_path-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/start_time-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/status-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/USER-group*
- */NXapm_paraprobe_results_spatstat/ENTRY/USER/address-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/USER/affiliation-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/USER/email-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/USER/name-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/USER/orcid-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/USER/orcid_platform-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/USER/role-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/USER/social_media_name-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/USER/social_media_platform-field*
- */NXapm_paraprobe_results_spatstat/ENTRY/USER/telephone_number-field*
- */NXapm_paraprobe_results_spatstat/ENTRY@version-attribute*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_results_spatstat.nxdl.xml

NXapm_paraprobe_results_surfacer

Status:

application definition, extends *NXobject*

Description:

Results of a paraprobe-surfacer tool run.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ions: The total number of ions in the reconstruction.

n_v_tri: The number of vertices of the alpha complex.

n_f_tri: The number of faces of the alpha complex.

n_f_tri_xdmf: The total number of XDMF values to represent all faces of triangles via XDMF.

n_f_tet_xdmf: The total number of XDMF values to represent all faces of tetrahedra via XDMF.

Groups cited:

NXcg_alpha_complex, *NXcg_face_list_data_structure*, *NXcg_tetrahedron_set*, *NXcg_triangle_set*, *NXcoordinate_system_set*, *NXcs_computer*, *NXcs_cpu*, *NXcs_filter_boolean_mask*, *NXcs_gpu*, *NXcs_io_obj*, *NXcs_io_sys*, *NXcs_mm_sys*, *NXcs_profiling_event*, *NXcs_profiling*, *NXentry*, *NXfabrication*, *NXprocess*, *NXtransformations*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_results_surfacer`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

start_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file was started, i.e. the paraprobe-tool executable started as a process.

end_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file were completed and the paraprobe-tool executable exited as a process.

config_filename: (required) *NX_CHAR*

The absolute path and name of the config file for this analysis.

@version: (required) *NX_CHAR*

At least SHA256 strong hash of the specific config_file for tracking provenance.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

status: (required) *NX_CHAR*

A statement whether the paraprobe-tool executable managed to process the analysis or failed prematurely.

This status is written to the results file after the end_time at which point the executable must not compute any analysis. Only when this status message is present and shows *success*, the user should consider the results. In all other cases it might be that the executable has terminated prematurely or another error occurred.

Any of these values: `success` | `failure`

USER: (recommended) *NXuser <=*

If used, contact information and eventually details of at least the person who performed this analysis. **name**: (required) *NX_CHAR <=*

affiliation: (recommended) *NX_CHAR <=*

address: (optional) *NX_CHAR <=*

email: (recommended) *NX_CHAR <=*

orcid: (recommended) *NX_CHAR <=*

orcid_platform: (recommended) *NX_CHAR <=*

telephone_number: (optional) *NX_CHAR <=*

role: (recommended) *NX_CHAR <=*

social_media_name: (optional) *NX_CHAR <=*

social_media_platform: (optional) *NX_CHAR <=*

COORDINATE_SYSTEM_SET: (required) *NXcoordinate_system_set*

Details about the coordinate system conventions used.

TRANSFORMATIONS: (required) *NXtransformations <=*

The individual coordinate systems which should be used. Field names should be prefixed with the following controlled terms indicating which individual coordinate system is described:

- paraprobe
- lab
- specimen
- laser
- leap
- detector
- recon

PROCESS: (optional) *NXprocess <=*

window: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies which of the ions in the dataset were analyzed. Computations of alpha complexes may have filtered this ion set further but this process is deterministic. **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of ions covered by the mask. The mask may be 0 for most.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} *<=*

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} *<=*

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

POINT_SET_WRAPPING: (optional) *NXprocess*

Paraprobe-surfacer can be used to load a ROI that is the entire or a sub-set of the ion point cloud. In the point_cloud_wrapping process the tool computes a triangulated surface mesh which encloses the ROI/point cloud. This mesh can be seen as a model for the edge of the dataset. Different algorithms can be used with paraprobe-surfacer to create this mesh such as convex hulls, alpha-shapes as their generalization, or alpha wrappings. Ideally, the resulting mesh should be a watertight polyhedron. This polyhedron is not necessarily convex. For some algorithms there is no guarantee that the resulting mesh yields a watertight mesh. **alpha_complex**: (optional) *NXcg_alpha_complex*

dimensionality: (required) *NX_UINT* {units=*NX_UNITLESS*} *<=*

Any of these values: `2` | `3`

type: (required) *NX_CHAR* *<=*

Any of these values:

- `convex_hull`
- `alpha_shape`
- `alpha_wrapping`
- `other`
- `undefined`

mode: (required) *NX_CHAR* *<=*

Any of these values: `general` | `regularized`

alpha: (required) *NX_NUMBER* {units=*NX_LENGTH*} *<=*

offset: (optional) *NX_NUMBER* {units=*NX_LENGTH*} *<=*

window: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies exactly all those ions whose positions were considered when defining the filtered point set from which the alpha complex was then in fact computed. This window can be different to the entire window as irrelevant ions might have been filtered out to reduce the computational costs of the alpha complex analysis. **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of ions covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*} <=

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

triangle_set: (optional) *NXcg_triangle_set* <=

The set of triangles in the coordinate system paraprobe which discretizes the exterior surface of the alpha complex. **identifier_offset**: (required) *NX_INT* {units=*NX_UNITLESS*} <=

Integer which specifies the first index to be used for distinguishing triangles. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1].

is_watertight: (optional) *NX_BOOLEAN*

Do the triangles define a triangulated surface mesh which is watertight?

volume: (optional) *NX_FLOAT* {units=*NX_VOLUME*}

The volume which the triangulated surface mesh encloses provided that the mesh is watertight.

triangles: (required) *NXcg_face_list_data_structure* <=

dimensionality: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

number_of_vertices: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

Number of vertices.

number_of_faces: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

Number of faces.

vertex_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

face_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

vertices: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_v_tri, 3]) {units=*NX_LENGTH*}

faces: (required) *NX_UINT* (Rank: 2, Dimensions: [n_f_tri, 3]) {units=*NX_UNITLESS*}

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [n_f_tri_xdmf]) {units=*NX_UNITLESS*}

A list of as many tuples of XDMF topology key, XDMF number of vertices and a triple of vertex indices specifying each triangle. The total number of entries is n_f_tri * (1+1+3).

interior_tetrahedra: (optional) *NXcg_tetrahedron_set*

The set of tetrahedra which represent the interior volume of the complex if that is a closed 2-manifold. **identifier_offset**: (required) *NX_INT* {units=*NX_UNITLESS*} *<=*

Integer which specifies the first index to be used for distinguishing tetrahedra. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

volume: (optional) *NX_FLOAT* {units=*NX_VOLUME*}

The accumulated volume of all interior tetrahedra.

tetrahedra: (optional) *NXcg_face_list_data_structure <=*

number_of_vertices: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

Number of vertices.

number_of_faces: (required) *NX_POSINT* {units=*NX_UNITLESS*} *<=*

Number of faces.

vertex_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} *<=*

face_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} *<=*

vertices: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_v_tet, 3]) {units=*NX_LENGTH*}

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [n_f_tet_xdmf]) {units=*NX_UNITLESS*}

A list of as many tuples of XDMF topology key, XDMF number of vertices and a triple of vertex indices specifying each triangle. The total number of entries is n_f_tet * (1+1+4).

TRIANGLE_SET_WRAPPING: (optional) *NXprocess*

In the future we may want to wrap other primitives like triangles or polylines.

performance: (recommended) *NXcs_profiling*

- **current_working_directory**: (required) *NX_CHAR* <=
- **command_line_call**: (optional) *NX_CHAR* <=
- **start_time**: (recommended) *NX_DATE_TIME* <=
- **end_time**: (recommended) *NX_DATE_TIME* <=
- **total_elapsed_time**: (required) *NX_NUMBER* <=
- **number_of_processes**: (required) *NX_POSINT* <=
- **number_of_threads**: (required) *NX_POSINT* <=
- **number_of_gpus**: (required) *NX_POSINT* <=
- **CS_COMPUTER**: (recommended) *NXcs_computer* <=
  - **name**: (recommended) *NX_CHAR* <=
  - **operating_system**: (required) *NX_CHAR* <=
    - **@version**: (required) *NX_CHAR* <=
  - **uuid**: (optional) *NX_CHAR* <=
  - **CS_CPU**: (optional) *NXcs_cpu* <=
    - **name**: (optional) *NX_CHAR* <=
    - **FABRICATION**: (recommended) *NXfabrication* <=
      - **identifier**: (optional) *NX_CHAR* <=
      - **capabilities**: (optional) *NX_CHAR*
  - **CS_GPU**: (optional) *NXcs_gpu* <=
    - **name**: (optional) *NX_CHAR* <=
    - **FABRICATION**: (recommended) *NXfabrication* <=
      - **identifier**: (optional) *NX_CHAR* <=
      - **capabilities**: (optional) *NX_CHAR*
  - **CS_MM_SYS**: (optional) *NXcs_mm_sys* <=
    - **total_physical_memory**: (required) *NX_NUMBER* <=
  - **CS_IO_SYS**: (optional) *NXcs_io_sys* <=
    - **CS_IO_OBJ**: (required) *NXcs_io_obj* <=
      - **technology**: (required) *NX_CHAR* <=
      - **max_physical_capacity**: (required) *NX_NUMBER* <=
      - **name**: (optional) *NX_CHAR* <=
      - **FABRICATION**: (recommended) *NXfabrication* <=
        - **identifier**: (optional) *NX_CHAR* <=
        - **capabilities**: (optional) *NX_CHAR*
  - **CS_PROFILING_EVENT**: (required) *NXcs_profiling_event*

start_time: (optional) *NX_DATE_TIME* <=

end_time: (optional) *NX_DATE_TIME* <=

description: (required) *NX_CHAR* <=

elapsed_time: (required) *NX_NUMBER* <=

number_of_processes: (required) *NX_POSINT* <=

Specify if it was different from the number_of_processes in the NXcs_profiling super class.

number_of_threads: (required) *NX_POSINT* <=

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

number_of_gpus: (required) *NX_POSINT* <=

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

max_virtual_memory_snapshot: (recommended) *NX_NUMBER* <=

max_resident_memory_snapshot: (recommended) *NX_NUMBER* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_results_surfacer/ENTRY-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/analysis_description-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/config_filename-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/config_filename@version-attribute*
- */NXapm_paraprobe_results_surfacer/ENTRY/COORDINATE_SYSTEM_SET-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/definition-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/end_time-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/command_line_call-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_CPU-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_CPU/name-field*

- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_GPU-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_GPU/name-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/capabiliti field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/max_physical_capacity-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/name-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/technology-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_MM_SYS-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_MM_SYS/total_physical_memory-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/description-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/elapsed_time-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/end_time-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_resident_memory_s field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_sn field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_gpus-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_processes-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_threads-field*

- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/start_time-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/name-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/operating_system-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/operating_system@version-attribute*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/uuid-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/end_time-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/number_of_gpus-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/number_of_processes-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/number_of_threads-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/start_time-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/total_elapsed_time-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/alpha-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/dimensionality-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/interior_tetrahedra-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/interior_tetrahedra/identifier field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/interior_tetrahedra/tetrahedr group*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/interior_tetrahedra/tetrahedr field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/interior_tetrahedra/tetrahedr field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/interior_tetrahedra/tetrahedr field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/interior_tetrahedra/tetrahedr field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/interior_tetrahedra/tetrahedr field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/interior_tetrahedra/tetrahedr field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/interior_tetrahedra/volume-field*

- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/mode-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/offset-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/triangle_set-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/triangle_set/identifier_offset-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/triangle_set/is_watertight-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/triangle_set/triangles-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/triangle_set/triangles/dimens field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/triangle_set/triangles/face_id field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/triangle_set/triangles/faces-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/triangle_set/triangles/number field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/triangle_set/triangles/number field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/triangle_set/triangles/vertex_ field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/triangle_set/triangles/vertices field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/triangle_set/triangles/xdmf_t field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/triangle_set/volume-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/type-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/window-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/window/bitdepth-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/window/mask-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/POINT_SET_WRAPPING/alpha_complex/window/number_of_ions-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/TRIANGLE_SET_WRAPPING-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/window-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/window/bitdepth-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/window/mask-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/PROCESS/window/number_of_ions-field*

- */NXapm_paraprobe_results_surfacer/ENTRY/program-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/program@version-attribute*
- */NXapm_paraprobe_results_surfacer/ENTRY/results_path-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/start_time-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/status-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/USER-group*
- */NXapm_paraprobe_results_surfacer/ENTRY/USER/address-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/USER/affiliation-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/USER/email-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/USER/name-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/USER/orcid-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/USER/orcid_platform-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/USER/role-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/USER/social_media_name-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/USER/social_media_platform-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/USER/telephone_number-field*
- */NXapm_paraprobe_results_surfacer/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_results_surfacer.nxdl.xml

NXapm_paraprobe_results_tessellator

Status:

application definition, extends *NXobject*

Description:

Results of a paraprobe-tessellator tool run.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ions: The total number of ions in the reconstruction.

n_f_xdmf: The total number of facets/polygons defining the tessellation.

Groups cited:
NXcg_face_list_data_structure, *NXcg_polyhedron_set*, *NXcoordinate_system_set*, *NXcs_computer*, *NXcs_cpu*, *NXcs_filter_boolean_mask*, *NXcs_gpu*, *NXcs_io_obj*, *NXcs_io_sys*, *NXcs_mm_sys*, *NXcs_profiling_event*, *NXcs_profiling*, *NXentry*, *NXfabrication*, *NXprocess*, *NXtransformations*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_results_tessellator`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

start_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file was started, i.e. the paraprobe-tool executable started as a process.

end_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file were completed and the paraprobe-tool executable exited as a process.

config_filename: (required) *NX_CHAR*

The absolute path and name of the config file for this analysis.

@version: (required) *NX_CHAR*

At least SHA256 strong hash of the specific config_file for tracking provenance.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

status: (required) *NX_CHAR*

A statement whether the paraprobe-tool executable managed to process the analysis or failed prematurely.

This status is written to the results file after the end_time at which point the executable must not compute any analysis. Only when this status message is present and shows *success*, the user should consider the results. In all other cases it might be that the executable has terminated prematurely or another error occurred.

Any of these values: `success` | `failure`

USER: (recommended) *NXuser <=*

If used, contact information and eventually details of at least the person who performed this analysis. **name**: (required) *NX_CHAR* <=

affiliation: (recommended) *NX_CHAR* <=

address: (optional) *NX_CHAR* <=

email: (recommended) *NX_CHAR* <=

orcid: (recommended) *NX_CHAR* <=

orcid_platform: (recommended) *NX_CHAR* <=

telephone_number: (optional) *NX_CHAR* <=

role: (recommended) *NX_CHAR* <=

social_media_name: (optional) *NX_CHAR* <=

social_media_platform: (optional) *NX_CHAR* <=

COORDINATE_SYSTEM_SET: (required) *NXcoordinate_system_set*

Details about the coordinate system conventions used.

TRANSFORMATIONS: (required) *NXtransformations* <=

The individual coordinate systems which should be used. Field names should be prefixed with the following controlled terms indicating which individual coordinate system is described:

- paraprobe
- lab
- specimen
- laser
- leap
- detector
- recon

PROCESS: (optional) *NXprocess* <=

voronoi_tessellation: (required) *NXprocess*

The tool can be used to compute a Voronoi tessellation the entire or a sub-set of the reconstruction. The point cloud in the ROI is wrapped into a tight axis-aligned bounding box. The tool detects if Voronoi cells make contact with the walls of this bounding box. The tessellation is computed without periodic boundary conditions. **window**: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies which of the ions in the dataset were analyzed. **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of ions covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*} <=

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

wall_contact_global: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies which of points have Voronoi cells that are truncated by the global axis-aligned bounding box, i.e. boundaries of the threads are ignored. **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of points covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} <=

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

wall_contact_left: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies which of points have Voronoi cells that are truncated by a specific wall of the axis-aligned bounding box. The left wall has the negative x axis of the paraprobe coordinate system as the outer unit normal. **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of points covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} <=

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

wall_contact_right: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies which of points have Voronoi cells that are truncated by a specific wall of the axis-aligned bounding box. The right wall has the positive x axis of the paraprobe coordinate system as the outer unit normal. **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of points covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} <=

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

wall_contact_front: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies which of points have Voronoi cells that are truncated by a specific wall of the axis-aligned bounding box. The front wall has the negative y axis of the paraprobe coordinate system as the outer unit normal. **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of points covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} <=

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

wall_contact_rear: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies which of points have Voronoi cells that are truncated by a specific wall of the axis-aligned bounding box. The rear wall has the positive y axis of the paraprobe coordinate system as the outer unit normal. **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of points covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} *<=*

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} *<=*

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

wall_contact_bottom: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies which of points have Voronoi cells that are truncated by a specific wall of the axis-aligned bounding box. The left wall has the negative z axis of the paraprobe coordinate system as the outer unit normal. **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of points covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} *<=*

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} *<=*

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

wall_contact_top: (required) *NXcs_filter_boolean_mask*

A bitmask which identifies which of points have Voronoi cells that are truncated by a specific wall of the axis-aligned bounding box. The left wall has the positive z axis of the paraprobe coordinate system as the outer unit normal. **number_of_ions**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of points covered by the mask. The mask value for most may be 0.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} <=

The unsigned integer array representing the content of the mask. If padding is used the padded bits are set to 0. The mask is for convenience always as large as the entire dataset as it will be stored compressed anyway. The convenience feature with this is that then the mask can be decoded with numpy and mirrored against the evaporation_id array and one immediately can filter out all points that were used by the paraprobe. The length of the array adds to the next unsigned integer if the number of ions in the dataset is not an integer multiple of the bitdepth.

voronoi_cells: (optional) *NXcg_polyhedron_set*

dimensionality: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

Obligatory value: 3

cardinality: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

volume: (required) *NX_NUMBER* (Rank: 1, Dimensions: [i]) {units=*NX_VOLUME*} <=

Interior volume

process_identifier: (optional) *NX_POSINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

By which MPI process was the Voronoi cell computed.

thread_identifier: (optional) *NX_POSINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

By which OpenMP thread was the Voronoi cell computed.

number_of_faces: (optional) *NX_POSINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} <=

The number of faces for each polyhedron. Faces of adjoining polyhedra are counted for each polyhedron. This field can be used to interpret the array/field with the individual area values for each face.

identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

Integer which specifies the first index to be used for distinguishing polyhedra. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} <=

Integer used to distinguish polyhedra for explicit indexing.

polyhedra: (optional) *NXcg_face_list_data_structure* <=

A simple approach to describe the entire set of polyhedra when the main intention is to store the shape of the polyhedra for visualization. **dimensionality**: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

number_of_vertices: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

Number of vertices.

number_of_faces: (required) *NX_POSINT* {units=*NX_UNITLESS*} <=

Number of faces.

vertex_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

face_identifier_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

vertices: (required) *NX_FLOAT* (Rank: 2, Dimensions: [i, 3]) {units=*NX_LENGTH*}

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

A sequence of always first an XDMF topology type key, followed by the XDMF number of vertices of the polygon, followed by the vertex identifier which define the facet polygon. First we store the polygon of the first facet of the first cell, then the second facet of the first cell, until the last facet of the first cell, followed by the first facet of the second cell, and so on and so forth.

xdmf_cell_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [n_f_xdmf]) {units=*NX_UNITLESS*}

A sequence of cell identifier so that each facet is associated with its cell because of which it is then possible to segment out cells three-dimensionally based on cell i.e. evaporation_id.

performance: (recommended) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* <=

command_line_call: (optional) *NX_CHAR* <=

start_time: (recommended) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

total_elapsed_time: (required) *NX_NUMBER* <=

number_of_processes: (required) *NX_POSINT* <=

number_of_threads: (required) *NX_POSINT* <=

number_of_gpus: (required) *NX_POSINT* <=

CS_COMPUTER: (recommended) *NXcs_computer* <=

- **name**: (recommended) *NX_CHAR* <=
- **operating_system**: (required) *NX_CHAR* <=
  - **@version**: (required) *NX_CHAR* <=
- **uuid**: (optional) *NX_CHAR* <=
- **CS_CPU**: (optional) *NXcs_cpu* <=
  - **name**: (optional) *NX_CHAR* <=
  - **FABRICATION**: (recommended) *NXfabrication* <=
    - **identifier**: (optional) *NX_CHAR* <=
    - **capabilities**: (optional) *NX_CHAR*
- **CS_GPU**: (optional) *NXcs_gpu* <=
  - **name**: (optional) *NX_CHAR* <=
  - **FABRICATION**: (recommended) *NXfabrication* <=
    - **identifier**: (optional) *NX_CHAR* <=
    - **capabilities**: (optional) *NX_CHAR*
- **CS_MM_SYS**: (optional) *NXcs_mm_sys* <=
  - **total_physical_memory**: (required) *NX_NUMBER* <=
- **CS_IO_SYS**: (optional) *NXcs_io_sys* <=
  - **CS_IO_OBJ**: (required) *NXcs_io_obj* <=
    - **technology**: (required) *NX_CHAR* <=
    - **max_physical_capacity**: (required) *NX_NUMBER* <=
    - **name**: (optional) *NX_CHAR* <=
    - **FABRICATION**: (recommended) *NXfabrication* <=
      - **identifier**: (optional) *NX_CHAR* <=
      - **capabilities**: (optional) *NX_CHAR*
- **CS_PROFILING_EVENT**: (required) *NXcs_profiling_event*
  - **start_time**: (optional) *NX_DATE_TIME* <=
  - **end_time**: (optional) *NX_DATE_TIME* <=
  - **description**: (required) *NX_CHAR* <=
  - **elapsed_time**: (required) *NX_NUMBER* <=
  - **number_of_processes**: (required) *NX_POSINT* <=

Specify if it was different from the number_of_processes in the NXcs_profiling super class.

number_of_threads: (required) *NX_POSINT <=*

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

number_of_gpus: (required) *NX_POSINT <=*

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

max_virtual_memory_snapshot: (recommended) *NX_NUMBER <=*

max_resident_memory_snapshot: (recommended) *NX_NUMBER <=*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_results_tessellator/ENTRY-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/analysis_description-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/config_filename-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/config_filename@version-attribute*
- */NXapm_paraprobe_results_tessellator/ENTRY/COORDINATE_SYSTEM_SET-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/definition-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/end_time-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/command_line_call-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_CPU-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_CPU/name-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_GPU-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/capabilities-field*

- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_GPU/name-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_IO_SYS-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/capabil field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identific field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/max_physical_capacity-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/name-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/technology-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_MM_SYS-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_MM_SYS/total_physical_memory-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/description-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/elapsed_time-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/end_time-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_resident_memory field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_ field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_gpus-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_processes-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_threads-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/start_time-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/name-field*

- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/operating_system-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/operating_system@version-attribute*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/uuid-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/end_time-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/number_of_gpus-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/number_of_processes-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/number_of_threads-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/start_time-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/total_elapsed_time-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/cardinality-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/dimensionality-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/identifier-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/identifier_offset-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/number_of_faces-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/polyhedra-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/polyhedra/dimensionality-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/polyhedra/face_identifier_offset-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/polyhedra/number_of_faces-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/polyhedra/number_of_vertices-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/polyhedra/vertex_identifier_offset-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/polyhedra/vertices-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/polyhedra/xdmf_cell_identifier-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/polyhedra/xdmf_topology-field*

- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/process_identifier-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/thread_identifier-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/voronoi_cells/volume-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_bottom-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_bottom/bitdepth-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_bottom/mask-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_bottom/number_of_ions-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_front-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_front/bitdepth-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_front/mask-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_front/number_of_ions-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_global-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_global/bitdepth-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_global/mask-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_global/number_of_ions-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_left-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_left/bitdepth-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_left/mask-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_left/number_of_ions-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_rear-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_rear/bitdepth-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_rear/mask-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_rear/number_of_ions-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_right-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_right/bitdepth-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_right/mask-field*

- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_right/number_of_ions-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_top-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_top/bitdepth-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_top/mask-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/wall_contact_top/number_of_ions-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/window-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/window/bitdepth-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/window/mask-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/PROCESS/voronoi_tessellation/window/number_of_ions-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/program-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/program@version-attribute*
- */NXapm_paraprobe_results_tessellator/ENTRY/results_path-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/start_time-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/status-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/USER-group*
- */NXapm_paraprobe_results_tessellator/ENTRY/USER/address-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/USER/affiliation-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/USER/email-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/USER/name-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/USER/orcid-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/USER/orcid_platform-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/USER/role-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/USER/social_media_name-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/USER/social_media_platform-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/USER/telephone_number-field*
- */NXapm_paraprobe_results_tessellator/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_results_tessellator.nxdl.xml

NXapm_paraprobe_results_transcoder

Status:

application definition, extends *NXobject*

Description:

Results of a paraprobe-transcoder tool run.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ions: The total number of ions in the reconstruction.

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 intervals mapped on this ion type.

n_topology: Total number of integers in the supplementary XDMF topology array.

n_combinatorics: Number of ions probed in the combinatorial analysis of the charge states

Groups cited:
NXcoordinate_system_set, *NXcs_computer*, *NXcs_cpu*, *NXcs_gpu*, *NXcs_io_obj*, *NXcs_io_sys*, *NXcs_mm_sys*, *NXcs_profiling_event*, *NXcs_profiling*, *NXentry*, *NXfabrication*, *NXinstrument*, *NXion*, *NXprocess*, *NXtransformations*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXapm_paraprobe_results_transcoder`

program: (required) *NX_CHAR*

Given name of the program/software/tool with which this NeXus (configuration) file was generated. **@version**: (required) *NX_CHAR*

Ideally program version plus build number, or commit hash or description of ever persistent resources where the source code of the program and build instructions can be found so that the program can be configured ideally in such a manner that the result of this computational process is recreatable in the same deterministic manner.

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

start_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file was started, i.e. the paraprobe-tool executable started as a process.

end_time: (required) *NX_DATE_TIME <=*

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis behind this results file were completed and the paraprobe-tool executable exited as a process.

config_filename: (required) *NX_CHAR*

The absolute path and name of the config file for this analysis.

@version: (required) *NX_CHAR*

At least SHA256 strong hash of the specific config_file for tracking provenance.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

status: (required) *NX_CHAR*

A statement whether the paraprobe-tool executable managed to process the analysis or failed prematurely.

This status is written to the results file after the end_time at which point the executable must not compute any analysis. Only when this status message is present and shows *success*, the user should consider the results. In all other cases it might be that the executable has terminated prematurely or another error occurred.

Any of these values: `success` | `failure`

USER: (recommended) *NXuser <=*

If used, contact information and eventually details of at least the person who performed this analysis. **name**: (required) *NX_CHAR <=*

affiliation: (recommended) *NX_CHAR <=*

address: (optional) *NX_CHAR <=*

email: (recommended) *NX_CHAR <=*

orcid: (recommended) *NX_CHAR <=*

orcid_platform: (recommended) *NX_CHAR <=*

telephone_number: (optional) *NX_CHAR <=*

role: (recommended) *NX_CHAR <=*

social_media_name: (optional) *NX_CHAR <=*

social_media_platform: (optional) *NX_CHAR <=*

COORDINATE_SYSTEM_SET: (required) *NXcoordinate_system_set*

Details about the coordinate system conventions used.

TRANSFORMATIONS: (required) *NXtransformations <=*

The individual coordinate systems which should be used. Field names should be prefixed with the following controlled terms indicating which individual coordinate system is described:

- paraprobe
- lab
- specimen
- laser
- leap
- detector
- recon

visualization: (recommended) *NXprocess* <=

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [n_topology]) {units=*NX_UNITLESS*}

An array of triplets of integers which can serve as a supplementary array for Paraview to display the reconstruction. The XDMF datatype is here 1, the number of primitives 1 per triplet, the last integer in each triplet is the identifier of each point starting from zero.

atom_probe: (required) *NXinstrument* <=

On a mid term perspective we would like to evolve the paraprobe-toolbox to an implementation stage where it works exclusively with completely provenance-tracked formats for both the configuration of the workflow step and/or analysis with each tool and also for the output of these analyses in the form of so-called tool-specific results files. Currently the Hierarchical Data Format 5 (HDF5) is used to store such data.

Different file formats can be used to inject reconstructed datasets and ranging definitions into the toolbox. Traditionally, these are the POS, ePOS, and APT files with the tomographic reconstruction and other metadata and RNG and RRNG file formats for the ranging definitions how mass-to-charge state-ratio values map on (molecular) ion types. Such input should be injected via specific NeXus/HDF5 files which are documented in compliance with the NXapm application definition.

So far the paraprobe-toolbox was used as a standalone tool. Therefore, it was not relevant during the development to focus on interoperability. Essentially paraprobe-transcoder was used as a parser to transcode data in the above-mentioned file formats into a paraprobe-specific representation. This transcoding should become deprecated. Here we describe steps we have taken into this direction.

With the work in the FAIRmat project and the desire to make the paraprobe- toolbox also accessible as a cloud-computing capable service in the Nomad Remote Tools Hub (NORTH) the topic of interoperability became more important and eventually the NXapm application definition was proposed. NORTH is a GUI and related service in a NOMAD OASIS instance which allows to spawn preconfigured docker containers via JupyterHub. Currently, NORTH includes the so-called apm container. A container with tools specific for analyzing data from atom probe microscopy as well as processing of point cloud and mesh data.

The NXapm application definition and related implementation work within NOMAD OASIS enabled users to parse content of POS, ePOS, APT, RNG, and RRNG files, surplus key metadata from vendor-agnostic electronic lab notebook solutions directly into NOMAD OASIS via the uploads section. The process is automated and yields an NXapm-compliant NeXus/HDF5 file inside the uploads section in return.

With these improvements made there is no longer a need for - at least the users of a NOMAD OASIS and NORTH instance to use the deprecated PARAPROBE.Transcoder.Results.*.h5 files. Ideally, paraprobe should automatically detect that the input can now be an NXapm-compliant NeXus/HDF5 file and in response work with this file directly. To remain compliant with users however who do not have or do not wish to use a NOMAD OASIS or NXapm or NeXus at all right now, the solution is as follows:

Calling the configuration stage of paraprobe-transcoder is always mandatory. It is always the first step of working with the toolbox. In this process the user defines the input files. These can either be nxs i.e. the NXapm/NeXus/ HDF5 file from e.g. the upload section, or such a file that was obtained from a colleague with a NOMAD OASIS instance. In all other cases, users can pass the reconstruction and ranging definitions using the traditional POS, ePOS, or APT and RNG or RRNG file formats respectively.

Based on which input the user delivers, the parmsetup-transcoder tool then creates a configuration file PARAPROBE.Transcoder.Config.SimID.*.nxs and informs the user whether the input was NeXus (and thus if all relevant input is already available) or whether the paraprobe-transcoder tool needs to be executed to convert the content of the vendor files first into a format which paraprobe can provenance track and understand. In the latter case, the PARAPROBE.Transcoder.Config.SimID.*.nxs file is used to communicate to all subsequently used tools from which files the tools can expect to find the reconstruction and ranging definitions.

All subsequent analysis steps start also with a tool-specific configuration. This configuration step reads in (among others) the PARAPROBE.Transcoder.Config.SimID.*.nxs file from which the configuration tool identifies automatically whether to read the reconstruction and ranging data from PARAPROBE.Transcoder.Results.SimID.*.h5 or directly the NXapm-compliant NeXus/HDF5 file that was created upon preparing the upload or the file shared from a colleague. This design removes the need for unnecessary copies of the data. Currently still though users should execute the transcoder step as it will generate a supplementary XDMF topology field with which the data in either the NeXus/HDF5 or the transcoded vendor files can be displayed using e.g. Paraview. For this purpose XDMF is used.

Of course ideally the APT community would at some point converge to use a common data exchange file format. To this end, AMETEK/Cameca's APT file format could be a good starting point but so far it is lacking a consistent way of how to store generalized ranging definitions and post-processing results. POS, ePOS, Rouen's ATO, as well as other so far used representations of data like CSV or text files have, to the best of our current knowledge, no concept of how to marry reconstruction and (optional) ranging data into one self-descriptive format.

This summarizes the rationale behind the current choices of the I/O for paraprobe. Furthermore, this summarizes also why the fundamental design of splitting an analysis always into steps of configuration (with parmsetup), task execution (with the respective C/C++ or Python tool of the toolbox), and post-processing (e.g. with autoreporter) is useful because it offers a clear description of provenance tracking. This is a necessary step to make atom probe microscopy data at all better aligned with the aims of the FAIR principles.

The internal organization of the data entries in the atom_probe group in this application definition for paraprobe-transcoder results files mirror the definitions of the NXapm for consistency reasons. **mass_to_charge_conversion**: (required) *NXprocess*

mass_to_charge: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_ANY*}

Mass-to-charge-state ratio values.

reconstruction: (required) *NXprocess*

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.

ranging: (required) *NXprocess*

peak_identification: (required) *NXprocess*

Details about how peaks, with taking into account error models, were interpreted as ion types or not. **ION**: (required) *NXion*

isotope_vector: (required) *NX_UINT <=*

nuclid_list: (recommended) *NX_UINT <=*

charge_state: (required) *NX_INT <=*

mass_to_charge_range: (required) *NX_FLOAT <=*

charge_model: (required) *NXprocess*

Details and results of the combinatorial analyses of this range definition to identify the charge_state for an ion. **charge_vector**: (required) *NX_UINT* (Rank: 1, Dimensions: [n_combinatorics]) {units=*NX_UNITLESS*}

Currently charge_state not charge!

isotope_matrix: (required) *NX_UINT* (Rank: 2, Dimensions: [n_combinatorics, n_ivec_max]) {units=*NX_UNITLESS*}

Specific isotopes building each candidate matching the range.

mass_vector: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_combinatorics]) {units=*NX_ANY*}

Accumulated mass of the isotopes in each candidate. Not corrected for quantum effects.

natural_abundance_product_vector: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_combinatorics]) {units=*NX_DIMENSIONLESS*}

Product of natural abundance of the isotopes per candidate.

min_abundance_product: (required) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Filter criterion on the product of the natural abundances computed from each isotope building the (molecular) ion. Such a filter can be used to reduce the number of possible molecular ions considered when trying to find a unique solution to the question which charge_state does a molecular ion within a given range and given combination of elements have.

min_half_life: (required) *NX_FLOAT* {units=*NX_TIME*}

Filter criterion on the minimum half life which all isotopes building the (molecular) ion need to have to consider the candidate. Such a filter can be used to reduce the number of possible molecular ions considered when trying to find a unique solution to the question which charge_state does a molecular ion within a given range and given combination of elements have.

sacrifice_isotopic_uniqueness: (required) *NX_BOOLEAN*

If the value is zero/false it means that non-unique solutions are accepted. These are solutions where multiple candidates differ in their isotopes but have the same charge.

performance: (required) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* <=

command_line_call: (optional) *NX_CHAR* <=

start_time: (recommended) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

total_elapsed_time: (required) *NX_NUMBER* <=

number_of_processes: (required) *NX_POSINT* <=

number_of_threads: (required) *NX_POSINT* <=

number_of_gpus: (required) *NX_POSINT* <=

CS_COMPUTER: (recommended) *NXcs_computer* <=

name: (recommended) *NX_CHAR* <=

operating_system: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

uuid: (optional) *NX_CHAR* <=

CS_CPU: (optional) *NXcs_cpu* <=

name: (optional) *NX_CHAR* <=

FABRICATION: (recommended) *NXfabrication* <=

identifier: (optional) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CS_GPU: (optional) *NXcs_gpu* <=

name: (optional) *NX_CHAR* <=

FABRICATION: (recommended) *NXfabrication* <=

identifier: (optional) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CS_MM_SYS: (optional) *NXcs_mm_sys* <=

total_physical_memory: (required) *NX_NUMBER* <=

CS_IO_SYS: (optional) *NXcs_io_sys* <=

CS_IO_OBJ: (required) *NXcs_io_obj* <=

technology: (required) *NX_CHAR* <=

max_physical_capacity: (required) *NX_NUMBER* <=

name: (optional) *NX_CHAR* <=

FABRICATION: (recommended) *NXfabrication* <=

identifier: (optional) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CS_PROFILING_EVENT: (required) *NXcs_profiling_event*

start_time: (optional) *NX_DATE_TIME* <=

end_time: (optional) *NX_DATE_TIME* <=

description: (required) *NX_CHAR* <=

elapsed_time: (required) *NX_NUMBER* <=

number_of_processes: (required) *NX_POSINT* <=

Specify if it was different from the number_of_processes in the NXcs_profiling super class.

number_of_threads: (required) *NX_POSINT* <=

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

number_of_gpus: (required) *NX_POSINT* <=

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

max_virtual_memory_snapshot: (recommended) *NX_NUMBER* <=

max_resident_memory_snapshot: (recommended) *NX_NUMBER* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_results_transcoder/ENTRY-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/analysis_description-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/analysis_identifier-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/mass_to_charge_conversion-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/mass_to_charge_conversion/mass_to_charge-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging/peak_identification-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging/peak_identification/ION-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging/peak_identification/ION/charge_model-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging/peak_identification/ION/charge_model/charge_vector-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging/peak_identification/ION/charge_model/isotope_matrix-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging/peak_identification/ION/charge_model/mass_vector-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging/peak_identification/ION/charge_model/min_abundance_pro field*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging/peak_identification/ION/charge_model/min_half_life-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging/peak_identification/ION/charge_model/natural_abundance_ field*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging/peak_identification/ION/charge_model/sacrifice_isotopic_u field*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging/peak_identification/ION/charge_state-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging/peak_identification/ION/isotope_vector-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging/peak_identification/ION/mass_to_charge_range-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/ranging/peak_identification/ION/nuclid_list-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/reconstruction-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/atom_probe/reconstruction/reconstructed_positions-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/config_filename-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/config_filename@version-attribute*
- */NXapm_paraprobe_results_transcoder/ENTRY/COORDINATE_SYSTEM_SET-group*

- */NXapm_paraprobe_results_transcoder/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/definition-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/end_time-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/command_line_call-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_CPU-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_CPU/name-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_GPU-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/capabilities-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/identifier-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_GPU/name-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_IO_SYS-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/capabi field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identifi field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/max_physical_capacity field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/name-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/technology-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_MM_SYS-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_MM_SYS/total_physical_memory-field*

- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/description-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/elapsed_time-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/end_time-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_resident_memory field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_ field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_gpus-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_processes-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_threads-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/start_time-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/name-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/operating_system-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/operating_system@version-attribute*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/uuid-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/current_working_directory-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/end_time-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/number_of_gpus-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/number_of_processes-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/number_of_threads-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/start_time-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/total_elapsed_time-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/program-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/program@version-attribute*
- */NXapm_paraprobe_results_transcoder/ENTRY/results_path-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/start_time-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/status-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/USER-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/USER/address-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/USER/affiliation-field*

- */NXapm_paraprobe_results_transcoder/ENTRY/USER/email-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/USER/name-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/USER/orcid-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/USER/orcid_platform-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/USER/role-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/USER/social_media_name-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/USER/social_media_platform-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/USER/telephone_number-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/visualization-group*
- */NXapm_paraprobe_results_transcoder/ENTRY/visualization/xdmf_topology-field*
- */NXapm_paraprobe_results_transcoder/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_results_transcoder.nxdl.xml

NXbeam_path

Status:

base class, extends *NXobject*

Description:

A beam path consisting of one or more optical elements.

NXbeam_path is used in NXopt to describe the beam path, i.e. the arrangement of optical elements between the excitation source and the sample, or between the sample and the detector unit.

To describe the order of the elements, use 'order(NXtransformations)', where each element's position points to the preceding element via '@depends_on'. Special case beam splitter: A beam splitter is a device which separates the beam into two or more beams. If such a device is part of the beam path use two or more NXbeam_paths to describe the beam paths after the beam splitter. In this case, in the dependency chain of the new beam paths, the first elements each point to the beam splitter, as this is the previous element.

Describe the relevant optical elements in the beam path by using the appropriate base classes. You may use as many elements as needed, also several elements of the same type as long as each element has its own name.

Symbols:

No symbol table

Groups cited:
NXaperture, *NXattenuator*, *NXbeam_splitter*, *NXdisk_chopper*, *NXfiber*, *NXfilter*, *NXgeometry*, *NXgrating*, *NXlens_opt*, *NXmirror*, *NXmonochromator*, *NXpinhole*, *NXpolarizer_opt*, *NXslit*, *NXsource*, *NXtransformations*, *NXwaveplate*, *NXxraylens*

Structure:

depends_on: (optional) *NX_CHAR*

Entry point of the dependency chain defined by the NXtransformations field, i.e. a link to the last element in the beam path. Example: /entry/instrument/beam_path/detector.

TRANSFORMATIONS: (optional) *NXtransformations*

Specify the order of the optical elements by defining a dependency chain. For each element, a '@depends_on' attribute should be used to state the position of the element in the beam path by pointing to the previous element. For the very first element, use the string "." instead. **AXISNAME**: (optional) *NX_NUMBER* {units=*NX_TRANSFORMATION*} *<=*

For each element in the beam path, one such field must exist with a '@depends_on' attribute defined to specify its position in the beam path. Note that also 'NXopt/ENTRY/INSTRUMENT/sample_stage' and windows ('NXopt/ENTRY/INSTRUMENT/sample_stage/entry_window' and 'NXopt/ENTRY/INSTRUMENT/sample_stage/exit_window') may be added to the dependency chain, i.e. may have an entry in this class even if they are not described in the beam path. ELEMENT is a place holder for the name of an optical beam path element. Note that the name of this field must be exactly the same as the element's field name. **@depends_on**: (optional) *NX_CHAR <=*

Add a link to the previous beam path element.

SOURCE: (optional) *NXsource*

Excitation source. One or more may be needed (e.g. two for a pump-probe setup with one pump and one probe beam). Depending on the source type, different properties are relevant, which are provided through the respective base class (e.g. use NXopt_source for lamps or lasers, NXchem_source for chemical reaction etc.). Some base classes are incomplete (NXchem_source, NXbio_source); the expertise of the respective communities is needed. **depends_on**: (optional) *NX_CHAR <=*

Use this field to point to the previous optical element.

type: (optional) *NX_CHAR <=*

Type of excitation source.

Any of these values:

- `semiconductor laser`
- `gas laser`
- `other laser`
- `lamp`
- `X-rays`
- `silicon carbide globar`
- `super continuum`
- `chemical reaction`
- `ultrasound`
- `sound`
- `living organism`
- `power supply`
- `electron source`
- `other`

lifespan: (optional) *NX_NUMBER* {units=*NX_TIME*}

Lifespan of the excitation (typically provided in hours).

measure_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

How many hours has the lamp been used?

excitation_wavelength: (optional) *NX_FLOAT* {units=*NX_ANY*}

Wavelengths or energy vector of the excitation source. This can be a single value or a spectrum, depending on the type of experiment. **@units**: (optional) *NX_CHAR*

Unit of wavelength or energy.

beam_profile: (optional) *NX_CHAR* (Rank: 2, Dimensions: [N_beam_profile_dim, N_beam_profile_points])

Two- or three-dimensional beam profile.

peak_power: (optional) *NX_FLOAT* {units=*NX_POWER*}

Power of one light pulse if the source is a pulsed source.

cw: (optional) *NX_BOOLEAN*

Is the excitation source continuous wave (CW)?

cw_power: (optional) *NX_FLOAT*

Power of CW beam.

bandwidth: (optional) *NX_FLOAT* {units=*NX_WAVELENGTH*}

FWHM bandwidth of the excitation source.

coherence_length: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Coherence length.

divergence: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

Divergence of the excitation beam.

PINHOLE: (optional) *NXpinhole*

Use this field to describe a simple pinhole (round geometry). Define its dimension using 'diameter'. For more complex geometries, 'NXaperture' should be used.

SLIT: (optional) *NXslit*

Use this field to describe a simple slit (rectangular geometry). Define its dimensions using 'x_gap' and 'y_gap'. For more complex geometries, 'NXaperture' should be used.

aperture_NUMBER: (optional) *NXaperture*

Use this field to describe an aperture. To specify a window, use the field 'window_NUMBER(NXaperture)'.

window_NUMBER: (optional) *NXaperture*

A window, e.g. an entry or exit window of a cryostat.

depends_on: (optional) *NX_CHAR* *<=*

Use this field to point to the previous optical element.

material: (optional) *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 material, decsribe here what it is.

thickness: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Thickness of the window

orientation_angle: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

Angle of the window normal (outer) vs. the substrate normal (similar to the angle of incidence).

reference_data_link: (optional) *NX_CHAR*

If reference data were measured add a link to the NeXus file where they are described.

MIRROR: (optional) *NXmirror*

filter_NUMBER: (optional) *NXfilter*

ATTENUATOR: (optional) *NXattenuator*

A device that reduces the intensity of a beam by attenuation.

attenuator_transmission: (optional) *NX_FLOAT* {units=*NX_DIMENSIONLESS*} *<=*

The transmitted intensity divided by the incident intensity.

attenuation: (optional) *NX_FLOAT* {units=*NX_ANY*}

Attenuation of the attenuator in dB.

@units: (optional) *NX_CHAR*

Unit of the measured data is not covered by NXDL units state here which unit was used.

APERTURE: (optional) *NXaperture*

Input and output aperture of the attenuator.

GEOMETRY: (optional) *NXgeometry*

Geometry (shape, size etc.) of the attenuator.

GRATING: (optional) *NXgrating*

A diffraction grating. Define relevant parameters in the corresponding fields, e.g. order of diffration (diffraction_order) or angular dispersion (angular_dispersion). **type**: (optional) *NX_CHAR*

Define the type of the grating.

angular_dispersion: (optional) *NX_FLOAT* {units=*NX_UNITLESS*}

Dispersion of the grating in nm/mm (or e.g. nm/mrad).

grooves: (optional) *NX_FLOAT* {units=*NX_PER_LENGTH*}

Number of grooves per mm.

blaze_wavelength: (optional) *NX_FLOAT* {units=*NX_WAVELENGTH*}

Blaze wavelength of the grating.

efficiency: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [N_spectrum]) {units=*NX_UNITLESS*}

Efficiency curve versus wavelength or energy.

spectrum: (optional) *NX_FLOAT* {units=*NX_ANY*}

Spectral values, e.g. wavelength or energy. Vector of length N_spectrum. **@units**: (optional) *NX_CHAR*

Unit of wavelength array (e.g. nanometer or Angstrom)

DISK_CHOPPER: (optional) *NXdisk_chopper*

A device blocking the beam in a temporal periodic pattern, e.g. a optical chopper wheel. Specify the frequency range using 'min_frequency' and 'max_frequency'. **min_frequency**: (optional) *NX_FLOAT* {units=*NX_FREQUENCY*}

Minimum frequency in Hertz.

max_frequency: (optional) *NX_FLOAT* {units=*NX_FREQUENCY*}

Maximum frequency in Hertz.

frequency_resolution: (optional) *NX_FLOAT* {units=*NX_FREQUENCY*}

Frequency resolution in Hertz.

MONOCHROMATOR: (optional) *NXmonochromator*

A monochromator or spectrometer.

spectrum: (optional) *NX_FLOAT* {units=*NX_ANY*}

Spectral values of the monochromator, e.g. wavelength or energy values used for the measurement. **@units**: (optional) *NX_CHAR*

Unit of wavelength array (e.g. nanometer or Angstrom)

spectral_resolution: (optional) *NX_FLOAT* {units=*NX_WAVENUMBER*}

Spectral resolution of the instrument.

GRATING: (optional) *NXgrating* *<=*

Diffraction grating. 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_FLOAT* {units=*NX_DIMENSIONLESS*}

Dispersion of the grating in nm/mm.

grating_wavelength_min: (optional) *NX_FLOAT* {units=*NX_WAVELENGTH*}

Minimum wavelength of the grating.

grating_wavelength_max: (optional) *NX_FLOAT* {units=*NX_WAVELENGTH*}

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_FLOAT* {units=*NX_LENGTH*}

If slit width was not fixed, define the maximum slit width.

XRAYLENS: (optional) *NXxraylens*

POLARIZER_OPT: (optional) *NXpolarizer_opt*

BEAM_SPLITTER: (optional) *NXbeam_splitter*

WAVEPLATE: (optional) *NXwaveplate*

LENS_OPT: (optional) *NXlens_opt*

FIBER: (optional) *NXfiber*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXbeam_path/aperture_NUMBER-group*
- */NXbeam_path/ATTENUATOR-group*
- */NXbeam_path/ATTENUATOR/APERTURE-group*
- */NXbeam_path/ATTENUATOR/attenuation-field*
- */NXbeam_path/ATTENUATOR/attenuation@units-attribute*
- */NXbeam_path/ATTENUATOR/attenuator_transmission-field*
- */NXbeam_path/ATTENUATOR/GEOMETRY-group*
- */NXbeam_path/BEAM_SPLITTER-group*
- */NXbeam_path/depends_on-field*
- */NXbeam_path/DISK_CHOPPER-group*
- */NXbeam_path/DISK_CHOPPER/frequency_resolution-field*
- */NXbeam_path/DISK_CHOPPER/max_frequency-field*
- */NXbeam_path/DISK_CHOPPER/min_frequency-field*
- */NXbeam_path/FIBER-group*
- */NXbeam_path/filter_NUMBER-group*
- */NXbeam_path/GRATING-group*
- */NXbeam_path/GRATING/angular_dispersion-field*
- */NXbeam_path/GRATING/blaze_wavelength-field*
- */NXbeam_path/GRATING/efficiency-field*

- */NXbeam_path/GRATING/grooves-field*
- */NXbeam_path/GRATING/spectrum-field*
- */NXbeam_path/GRATING/spectrum@units-attribute*
- */NXbeam_path/GRATING/type-field*
- */NXbeam_path/LENS_OPT-group*
- */NXbeam_path/MIRROR-group*
- */NXbeam_path/MONOCHROMATOR-group*
- */NXbeam_path/MONOCHROMATOR/GRATING-group*
- */NXbeam_path/MONOCHROMATOR/GRATING/angular_dispersion-field*
- */NXbeam_path/MONOCHROMATOR/GRATING/grating_wavelength_max-field*
- */NXbeam_path/MONOCHROMATOR/GRATING/grating_wavelength_min-field*
- */NXbeam_path/MONOCHROMATOR/SLIT-group*
- */NXbeam_path/MONOCHROMATOR/SLIT/fixed_slit-field*
- */NXbeam_path/MONOCHROMATOR/SLIT/max_gap-field*
- */NXbeam_path/MONOCHROMATOR/spectral_resolution-field*
- */NXbeam_path/MONOCHROMATOR/spectrum-field*
- */NXbeam_path/MONOCHROMATOR/spectrum@units-attribute*
- */NXbeam_path/PINHOLE-group*
- */NXbeam_path/POLARIZER_OPT-group*
- */NXbeam_path/SLIT-group*
- */NXbeam_path/SOURCE-group*
- */NXbeam_path/SOURCE/bandwidth-field*
- */NXbeam_path/SOURCE/beam_profile-field*
- */NXbeam_path/SOURCE/coherence_length-field*
- */NXbeam_path/SOURCE/cw-field*
- */NXbeam_path/SOURCE/cw_power-field*
- */NXbeam_path/SOURCE/depends_on-field*
- */NXbeam_path/SOURCE/divergence-field*
- */NXbeam_path/SOURCE/excitation_wavelength-field*
- */NXbeam_path/SOURCE/excitation_wavelength@units-attribute*
- */NXbeam_path/SOURCE/lifespan-field*
- */NXbeam_path/SOURCE/measure_time-field*
- */NXbeam_path/SOURCE/peak_power-field*
- */NXbeam_path/SOURCE/type-field*
- */NXbeam_path/TRANSFORMATIONS-group*
- */NXbeam_path/TRANSFORMATIONS/AXISNAME-field*

- */NXbeam_path/TRANSFORMATIONS/AXISNAME@depends_on-attribute*
- */NXbeam_path/WAVEPLATE-group*
- */NXbeam_path/window_NUMBER-group*
- */NXbeam_path/window_NUMBER/depends_on-field*
- */NXbeam_path/window_NUMBER/material-field*
- */NXbeam_path/window_NUMBER/orientation_angle-field*
- */NXbeam_path/window_NUMBER/other_material-field*
- */NXbeam_path/window_NUMBER/reference_data_link-field*
- */NXbeam_path/window_NUMBER/thickness-field*
- */NXbeam_path/XRAYLENS-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXbeam_path.nxdl.xml

NXbeam_splitter

Status:

base class, extends *NXobject*

Description:

A beam splitter, i.e. a device splitting the light into two or more beams.

Information about types and properties of beam splitters is provided e.g. [here](https://www.rp-photonics.com/beam_splitters.html).

Use two or more NXbeam_paths to describe the beam paths after the beam splitter. In the dependency chain of the new beam paths, the first elements each point to this beam splitter, as this is the previous element.

Symbols:

N_spectrum: Length of the spectrum vector (e.g. wavelength or energy) for which the refractive index of the beam splitter material and/or coating is defined.

N_spectrum_RT: Length of the spectrum vector (e.g. wavelength or energy) for which the reflectance or transmission of the beam splitter is given.

N_shapepar: Number of parameters needed do descripe the shape of the beam splitter.

N_objects: Number of objects the beam splitter is made up of.

N_outputs: Number of outputs, i.e. number of paths the beam takes after being split by the beam splitter.

Groups cited:
NXdata, *NXsample*, *NXshape*

Structure:

type: (optional) *NX_CHAR*

Specify the beam splitter type (e.g. dielectric mirror, pellicle, dichroic mirror etc.). Shape (e.g. prism, plate, cube) and dimension should be described in 'geometry'. Define if the beam splitter is polarizing or not in the field 'polarizing(NX_BOOLEAN)'.

Any of these values:

- `dichroic mirror`
- `dielectric mirror`
- `metal-coated mirror`
- `Nicol prism`
- `Glan-Thompson prism`
- `pellicle mirror`
- `Polka dot beam splitter`
- `fiber optic splitter`
- `other`

other_type: (optional) *NX_CHAR*

If you selected 'other' in 'type' use this field to specify which type of beam splitter was used.

polarizing: (optional) *NX_BOOLEAN*

Is the beam splitter polarizing?

multiple_outputs: (optional) *NX_BOOLEAN*

Does the beam splitter have multiple outputs (diffractive optical element), i.e. more than two outputs?

splitting_ratio: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [N_outputs]) {units=*NX_UNITLESS*}

Beam splitting ratio(s) for the various outputs (i.e. the paths of the beam after being split by the beam splitter). The order of the ratios must be consistent with the labels 1, 2, … N_outputs defined by the sketch in 'SHAPE/sketch', starting with 1.

clear_aperture: (optional) *NX_FLOAT* {units=*NX_UNITLESS*}

Clear aperture of the device (e.g. 90% of diameter for a disc, or 90% of length and height for square geometry).

wavelength_range: (recommended) *NX_FLOAT* (Rank: 1, Dimensions: [2]) {units=*NX_WAVELENGTH*}

Wavelength range for which the beam splitter is designed. Enter the minimum and maximum values of the wavelength range. Alternatively, or additionally, you may define the wavelength range for the coating in coating/wavelength_range_coating.

optical_loss: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [N_outputs]) {units=*NX_UNITLESS*}

Optical loss of the beam splitter for the various outputs (i.e. the paths of the beam after being split by the beam splitter). The order of the ratios must be consistent with the labels 1, 2, … N_outputs defined by the sketch in 'SHAPE/sketch', starting with 1.

incident_angle: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Optimized angle of incidence for the desired splitting ratio.

deflection_angle: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Angle of deflection corresponding to the optimized angle of incidence defined in incident_angle.

AOI_range: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [2]) {units=*NX_ANGLE*}

Range of the angles of incidence (AOI) for which the beam splitter can be operated. Specify the minimum and maximum angles of the range.

reflectance: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [N_spectrum_RT]) {units=*NX_UNITLESS*}

Reflectance of the beam splitter at given spectral values.

transmission: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [N_outputs, N_spectrum_RT]) {units=*NX_UNITLESS*}

Transmission at given spectral values for the various outputs (i.e. the paths of the beam after being split by the beam splitter). The order of the ratios must be consistent with the labels 1, 2, … N_outputs defined by the sketch in 'SHAPE/sketch', starting with 1.

SHAPE: (recommended) *NXshape*

Describe the geometry (shape, dimension etc.) of the beam splitter. Specify the dimensions in 'SHAPE/size'. A sketch of the device should be provided in the 'sketch(NXdata)' field to clarify (i) the shape and dimensions of the device, and (ii) the input and outputs (i.e. the direction of the incoming and outcoming (split) beams). **shape**: (optional) *NX_CHAR* *<=*

Describe the shape (plate, cube, wedged, prism etc.).

Any of these values:

- `cube`
- `cylinder`
- `plate`
- `prism`
- `wedged`
- `other`

other_shape: (optional) *NX_CHAR*

If you chose 'other' in 'shape' describe what it is.

size: (optional) *NX_CHAR* (Rank: 2, Dimensions: [N_objects, N_shapepar])

Physical extent of the beam splitter device. The beam splitter might be made up of one or more objects (NX_objects). The meaning and location of the axes used will vary according to the value of the 'shape' variable. 'N_shapepar' defines how many parameters:

- For 'cube' the parameters are (width, length).
- For 'cylinder' the parameters are (diameter, length).
- For 'plate' the parameters are (width, height, length).
- For 'prism' the parameters are (width, height, length).
- For 'wedged' the parameters are (width, height, shortest length). The wedge angle should be provided in 'SHAPE/wedge_angle'.
- For 'other' the parameters may be (A, B, C, …) with the labels defined in the sketch plotted in 'SHAPE/sketch'.

wedge_angle: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

Wedge angle if 'shape' is 'wedged'.

sketch: (optional) *NXdata*

Sketch of the beam splitter showing its geometry. The paths of the incoming and split beam should be illustrated and labelled (0 for the incoming beam, and 1, 2,…, N_outputs for the outputs (i.e. the split beam paths)).

substrate: (optional) *NXsample*

Substrate of the beam splitter. Describe the material of the substrate in substrate/substrate_material and provide its index of refraction in substrate/index_of_refraction_substrate, if known. **substrate_material**: (optional) *NX_CHAR*

Specify the material of the beam splitter. If the device has a coating it should be described in coating/coating_material. Is the material birefringent?

substrate_thickness: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [2]) {units=*NX_LENGTH*}

Thickness of the beam splitter substrate. Define the minimum and maximum thickness (for a wedged geomtry). For a homogeneous thickness (e.g. as in plate beam splitters) the minimum and maximum values are equal.

index_of_refration_substrate: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [2, N_spectrum]) {units=*NX_UNITLESS*}

Complex index of refraction of the beam splitter substrate. Specify at given spectral values (e.g. wavelength, energy, wavenumber etc.).

coating: (optional) *NXsample*

Is the beam splitter coated? If yes, specify the type and material of the coating and the spectral range for which it is designed. If known, you may also provide its index of refraction. For a beam splitter cube consisting of two prisms which are glued together, you may want to specify the the glue and the coatings of each prism. **coating_type**: (optional) *NX_CHAR*

Specify the coating type (e.g. dielectric, anti-reflection (AR), multilayer coating etc.).

coating_material: (optional) *NX_CHAR*

Specify the coating material.

coating_thickness: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Thickness of the coating.

wavelength_range_coating: (recommended) *NX_FLOAT* (Rank: 1, Dimensions: [2]) {units=*NX_WAVELENGTH*}

Wavelength range for which the coating is designed. Enter the minimum and maximum values of the wavelength range.

index_of_refraction_coating: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [2, N_spectrum]) {units=*NX_UNITLESS*}

Complex index of refraction of the coating. Specify at given spectral values (e.g. wavelength, energy, wavenumber etc.).

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXbeam_splitter/AOI_range-field*
- */NXbeam_splitter/clear_aperture-field*
- */NXbeam_splitter/coating-group*
- */NXbeam_splitter/coating/coating_material-field*
- */NXbeam_splitter/coating/coating_thickness-field*
- */NXbeam_splitter/coating/coating_type-field*
- */NXbeam_splitter/coating/index_of_refraction_coating-field*
- */NXbeam_splitter/coating/wavelength_range_coating-field*
- */NXbeam_splitter/deflection_angle-field*
- */NXbeam_splitter/incident_angle-field*
- */NXbeam_splitter/multiple_outputs-field*
- */NXbeam_splitter/optical_loss-field*
- */NXbeam_splitter/other_type-field*
- */NXbeam_splitter/polarizing-field*
- */NXbeam_splitter/reflectance-field*
- */NXbeam_splitter/SHAPE-group*
- */NXbeam_splitter/SHAPE/other_shape-field*
- */NXbeam_splitter/SHAPE/shape-field*
- */NXbeam_splitter/SHAPE/size-field*
- */NXbeam_splitter/SHAPE/sketch-group*
- */NXbeam_splitter/SHAPE/wedge_angle-field*
- */NXbeam_splitter/splitting_ratio-field*
- */NXbeam_splitter/substrate-group*
- */NXbeam_splitter/substrate/index_of_refration_substrate-field*
- */NXbeam_splitter/substrate/substrate_material-field*
- */NXbeam_splitter/substrate/substrate_thickness-field*
- */NXbeam_splitter/transmission-field*
- */NXbeam_splitter/type-field*
- */NXbeam_splitter/wavelength_range-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXbeam_splitter.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* (Rank: 1, Dimensions: [ncoeff]) {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* (Rank: 1, Dimensions: [ncal]) {units=*NX_ANY*}

A vector representing the axis after calibration, matching the data length

original_axis: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [ncal]) {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

NXcg_alpha_complex

Status:

base class, extends *NXobject*

Description:

Computational geometry description of alpha shapes or wrappings to primitives.

For details see:

- https://dx.doi.org/10.1109/TIT.1983.1056714 for 2D,
- https://dx.doi.org/10.1145/174462.156635 for 3D,
- https://dl.acm.org/doi/10.5555/871114 for weighted, and
- https://doc.cgal.org/latest/Alpha_shapes_3 for 3D implementation
- https://doc.cgal.org/latest/Manual/packages.html#PkgAlphaWrap3 for 3D wrap

in CGAL, the Computational Geometry Algorithms Library. As a starting point, we follow the conventions of the CGAL library.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: The dimensionality of the alpha shape, for now 2 or 3.

n_e: The number of edges.

n_f: The number of faces.

n_c: The number of cells.

Groups cited:
NXcg_point_set, *NXcg_tetrahedron_set*, *NXcg_triangle_set*

Structure:

dimensionality: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Any of these values: `2` | `3`

type: (optional) *NX_CHAR*

Specify which general type of alpha shape is computed. Using for now the CGAL terminology. Basic means (unweighted) alpha shapes. Alpha_wrapping means meshes created using alpha wrapping procedures.

Any of these values: `convex_hull` | `alpha_shape` | `alpha_wrapping`

mode: (optional) *NX_CHAR*

Specifically when computed with the CGAL, the mode specifies if singular faces are removed (regularized) of the alpha complex.

Any of these values: `general` | `regularized`

alpha: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The alpha, (radius of the alpha-sphere) parameter to be used for alpha shapes and alpha wrappings.

offset: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The offset distance parameter to be used in addition to alpha in the case of alpha_wrapping.

point_set: (optional) *NXcg_point_set*

Point cloud for which the alpha shape or wrapping should be computed.

triangle_set: (optional) *NXcg_triangle_set*

Triangle soup for which the alpha wrapping should be computed.

triangulation: (optional) *NXcg_triangle_set*

A meshed representation of the resulting shape.

interior_cells: (optional) *NXcg_tetrahedron_set*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_alpha_complex/alpha-field*
- */NXcg_alpha_complex/dimensionality-field*
- */NXcg_alpha_complex/interior_cells-group*
- */NXcg_alpha_complex/mode-field*
- */NXcg_alpha_complex/offset-field*
- */NXcg_alpha_complex/point_set-group*
- */NXcg_alpha_complex/triangle_set-group*
- */NXcg_alpha_complex/triangulation-group*
- */NXcg_alpha_complex/type-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_alpha_complex.nxdl.xml

NXcg_cylinder_set

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a set of cylinders in Euclidean space.

The members of the set can have different size. For each member the position of the center and the height is mandatory. The radius can either be defined in the radius field or by filling both the upper and the lower radius field. The latter case can be used to represent truncated cones.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

c: The cardinality of the set, i.e. the number of cylinders or cones.

Groups cited:
NXtransformations

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Obligatory value: 3

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing identifiers for cylinders. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c])

Integer used to distinguish members for explicit indexing.

center: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_LENGTH*}

The geometric center of each member.

height: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_LENGTH*}

A direction vector which is parallel to the cylinder/cone axis and whose magnitude is the height of the cylinder/cone.

radii: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_LENGTH*}

upper_cap_radius: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_LENGTH*}

The radius of the upper circular cap. This field, combined with lower_cap_radius can be used to describe (eventually truncated) circular cones.

lower_cap_radius: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_LENGTH*}

The radius of the upper circular cap. This field, combined with lower_cap_radius can be used to describe (eventually truncated) circular cones.

volume: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_VOLUME*}

Interior volume of each cylinder

lateral_surface_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*}

Lateral surface area

cap_surface_area: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 2]) {units=*NX_AREA*}

Area of the upper and the lower cap of each cylinder respectively.

surface_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*}

Cap and lateral surface area for each cylinder.

TRANSFORMATIONS: (optional) *NXtransformations*

Reference to or definition of a coordinate system with which the positions and directions are interpretable.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_cylinder_set/cap_surface_area-field*
- */NXcg_cylinder_set/cardinality-field*
- */NXcg_cylinder_set/center-field*
- */NXcg_cylinder_set/dimensionality-field*
- */NXcg_cylinder_set/height-field*
- */NXcg_cylinder_set/identifier-field*
- */NXcg_cylinder_set/identifier_offset-field*
- */NXcg_cylinder_set/lateral_surface_area-field*
- */NXcg_cylinder_set/lower_cap_radius-field*
- */NXcg_cylinder_set/radii-field*
- */NXcg_cylinder_set/surface_area-field*
- */NXcg_cylinder_set/TRANSFORMATIONS-group*
- */NXcg_cylinder_set/upper_cap_radius-field*
- */NXcg_cylinder_set/volume-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_cylinder_set.nxdl.xml

NXcg_ellipsoid_set

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a set of ellipsoids in Euclidean space.

Individual ellipsoids can have different half axes.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: The dimensionality, which has to be at least 2.

c: The cardinality of the set, i.e. the number of ellipses, or ellipsoids.

Groups cited:
NXtransformations

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing identifiers for ellipsoids. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Integer used to distinguish ellipsoids for explicit indexing.

center: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, d]) {units=*NX_LENGTH*}

Geometric center of the ellipsoids. This can be the center of mass. Dimensionality and cardinality of the point and ellipsoid set have to match. The identifier_offset and identifier field of NXcg_point_set do not need to be used as they should be same as the identifier_offset and the identifier for the ellipsoids.

half_axes_radius: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [d]) {units=*NX_LENGTH*}

If all ellipsoids in the set have the same half-axes.

half_axes_radii: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, d]) {units=*NX_LENGTH*}

In the case that ellipsoids have different radii use this field instead of half_axes_radius.

is_closed: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [c])

Are the ellipsoids closed or hollow?

volume: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_ANY*}

surface_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_ANY*}

orientation: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, d]) {units=*NX_DIMENSIONLESS*}

Direction unit vector which points along the largest half-axes.

TRANSFORMATIONS: (optional) *NXtransformations*

Reference to or definition of a coordinate system with which the positions and directions are interpretable.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_ellipsoid_set/cardinality-field*
- */NXcg_ellipsoid_set/center-field*
- */NXcg_ellipsoid_set/dimensionality-field*
- */NXcg_ellipsoid_set/half_axes_radii-field*
- */NXcg_ellipsoid_set/half_axes_radius-field*
- */NXcg_ellipsoid_set/identifier-field*
- */NXcg_ellipsoid_set/identifier_offset-field*
- */NXcg_ellipsoid_set/is_closed-field*
- */NXcg_ellipsoid_set/orientation-field*
- */NXcg_ellipsoid_set/surface_area-field*
- */NXcg_ellipsoid_set/TRANSFORMATIONS-group*
- */NXcg_ellipsoid_set/volume-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_ellipsoid_set.nxdl.xml

NXcg_face_list_data_structure

Status:

base class, extends *NXobject*

Description:

Computational geometry description of geometric primitives via a face and edge list.

Primitives must not be degenerated or self-intersect. Such descriptions of primitives are frequently used for triangles and polyhedra to store them on disk for visualization purposes. Although storage efficient, such a description is not well suited for topological and neighborhood queries of especially meshes that are built from primitives.

In this case, scientists may need a different view on the primitives which is better represented for instance with a half_edge_data_structure instance. The reason to split thus the geometric description of primitives, sets, and specifically meshes of primitives is to keep the structure simple enough for users without these computational geometry demands but also enable those more computational geometry savy users the storing of the additionally relevant data structure.

This is beneficial and superior over NXoff_geometry because for instance a half_edge_data_structure instance can be immediately use to reinstantiate the set without having to recompute the half_edge_structure from the vertex and face-list based representation and thus offer a more efficient route to serve applications where topological and graph-based operations are key.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: The dimensionality, which has to be at least 2.

n_v: The number of vertices.

n_e: The number of edges.

n_f: The number of faces.

n_total: The total number of vertices of all faces. Faces are polygons.

n_weinberg: The total number of Weinberg vector values of all faces.

Groups cited:
none

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Dimensionality.

number_of_vertices: (optional) *NX_POSINT* (Rank: 1, Dimensions: [n_f]) {units=*NX_UNITLESS*}

Array which specifies of how many vertices each face is built. Each entry represent the total number of vertices for face, irrespectively whether vertices are shared among faces/are unique or not.

number_of_edges: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Number of edges.

number_of_faces: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Number of faces.

vertex_identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing identifiers for vertices. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

edge_identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing identifiers for edges. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

face_identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing identifiers for faces. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

vertex_identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [n_v]) {units=*NX_UNITLESS*}

Integer used to distinguish vertices explicitly.

edge_identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [n_e]) {units=*NX_UNITLESS*}

Integer used to distinguish edges explicitly.

face_identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [n_f]) {units=*NX_UNITLESS*}

Integer used to distinguish faces explicitly.

vertices: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_v, d]) {units=*NX_LENGTH*}

Positions of the vertices.

Users are encouraged to reduce the vertices to unique set of positions and vertices as this supports a more efficient storage of the geometry data. It is also possible though to store the vertex positions naively in which case vertices_are_unique is likely False. Naively here means that one for example stores each vertex of a triangle mesh even though many vertices are shared between triangles and thus the positions of these vertices do not have to be duplicated.

edges: (optional) *NX_INT* (Rank: 2, Dimensions: [n_e, 2]) {units=*NX_UNITLESS*}

The edges are stored as a pairs of vertex identifier values.

faces: (optional) *NX_INT* (Rank: 1, Dimensions: [n_total]) {units=*NX_UNITLESS*}

Array of identifiers from vertices which describe each face.

The first entry is the identifier of the start vertex of the first face, followed by the second vertex of the first face, until the last vertex of the first face. Thereafter, the start vertex of the second face, the second vertex of the second face, and so on and so forth.

Therefore, summating over the number_of_vertices, allows to extract the vertex identifiers for the i-th face on the following index interval of the faces array: [$sum_i = 0}^{i = n-1}$, $sum_{i=0}^{i = n}$].

vertices_are_unique: (optional) *NX_BOOLEAN*

If true indicates that the vertices are all placed at different positions and have different identifiers, i.e. no points overlap or are counted twice.

edges_are_unique: (optional) *NX_BOOLEAN*

If true indicates that no edge is stored twice. Users are encouraged to consider and use the half_edge_data_structure instead as this will work towards achieving a cleaner graph-based description if relevant and possible.

faces_are_unique: (optional) *NX_BOOLEAN*

winding_order: (optional) *NX_INT* (Rank: 1, Dimensions: [n_f]) {units=*NX_UNITLESS*}

Specifies for each face which winding order was used if any:

- 0 - undefined
- 1 - counter-clockwise (CCW)
- 2 - clock-wise (CW)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_face_list_data_structure/dimensionality-field*
- */NXcg_face_list_data_structure/edge_identifier-field*
- */NXcg_face_list_data_structure/edge_identifier_offset-field*
- */NXcg_face_list_data_structure/edges-field*
- */NXcg_face_list_data_structure/edges_are_unique-field*
- */NXcg_face_list_data_structure/face_identifier-field*
- */NXcg_face_list_data_structure/face_identifier_offset-field*
- */NXcg_face_list_data_structure/faces-field*
- */NXcg_face_list_data_structure/faces_are_unique-field*
- */NXcg_face_list_data_structure/number_of_edges-field*
- */NXcg_face_list_data_structure/number_of_faces-field*
- */NXcg_face_list_data_structure/number_of_vertices-field*
- */NXcg_face_list_data_structure/vertex_identifier-field*
- */NXcg_face_list_data_structure/vertex_identifier_offset-field*
- */NXcg_face_list_data_structure/vertices-field*
- */NXcg_face_list_data_structure/vertices_are_unique-field*
- */NXcg_face_list_data_structure/winding_order-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_face_list_data_structure.nxdl.xml

NXcg_geodesic_mesh

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a geodesic mesh.

People from geodesic/surveyors will likely have specific demands and different views about what should be included in such a base class, given that nested geodesic meshes are a key component of climate modelling tools. For now we propose to use this base class as a container to organize metadata and data related to geodesic meshes.

Specifically an instance of this base class should detail the rule set how how the geodesic (surface) mesh was instantiated as there are many possibilities. A geodesic surface mesh is in this sense a triangulated surface mesh with metadata. For additional details as an introduction into the topic see e.g.:

- E. S. Popko and C. J. Kitrick

Here, especially the section on subdivision schemes is relevant.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXcg_triangulated_surface_mesh, *NXtransformations*

Structure:

TRANSFORMATIONS: (optional) *NXtransformations*

Reference to or definition of a coordinate system with which the positions and directions are interpretable.

CG_TRIANGULATED_SURFACE_MESH: (optional) *NXcg_triangulated_surface_mesh*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_geodesic_mesh/CG_TRIANGULATED_SURFACE_MESH-group*
- */NXcg_geodesic_mesh/TRANSFORMATIONS-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_geodesic_mesh.nxdl.xml

NXcg_grid

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a Wigner-Seitz cell grid in Euclidean space.

Frequently convenient three-dimensional grids with cubic cells are used. Exemplar applications are spectral-solver based crystal plasticity and stencil methods like phase-field or cellular automata.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: The dimensionality of the grid.

c: The cardinality or total number of cells/grid points.

n_b: Number of boundaries of the bounding box or primitive to the grid.

Groups cited:
NXcg_polyhedron_set, *NXtransformations*

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Any of these values: 1 | 2 | 3

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

origin: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [d])

symmetry: (optional) *NX_CHAR*

The symmetry of the lattice defining the shape of the unit cell.

Obligatory value: `cubic`

cell_dimensions: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [d]) {units=*NX_LENGTH*}

The unit cell dimensions using crystallographic notation.

extent: (optional) *NX_POSINT* (Rank: 1, Dimensions: [d]) {units=*NX_UNITLESS*}

Number of unit cells along each of the d unit vectors. The total number of cells, or grid points has to be the cardinality. If the grid has an irregular number of grid positions in each direction, as it could be for instance the case of a grid where all grid points outside some masking primitive are removed, this extent field should not be used. Instead use the coordinate field.

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing identifiers for cells. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Integer used to distinguish cells for explicit indexing.

position: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, d]) {units=*NX_LENGTH*}

Position of each cell in Euclidean space.

coordinate: (optional) *NX_INT* (Rank: 2, Dimensions: [c, d]) {units=*NX_DIMENSIONLESS*}

Coordinate of each cell with respect to the discrete grid.

number_of_boundaries: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

How many distinct boundaries are distinguished? Most grids discretize a cubic or cuboidal region. In this case six sides can be distinguished, each making an own boundary.

boundaries: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_b])

Name of domain boundaries of the simulation box/ROI e.g. left, right, front, back, bottom, top.

boundary_conditions: (optional) *NX_INT* (Rank: 1, Dimensions: [n_b]) {units=*NX_UNITLESS*}

The boundary conditions for each boundary:

0 - undefined 1 - open 2 - periodic 3 - mirror 4 - von Neumann 5 - Dirichlet

TRANSFORMATIONS: (optional) *NXtransformations*

Reference to or definition of a coordinate system with which the positions and directions are interpretable.

bounding_box: (optional) *NXcg_polyhedron_set*

A tight bounding box or sphere or bounding primitive about the grid.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_grid/boundaries-field*
- */NXcg_grid/boundary_conditions-field*
- */NXcg_grid/bounding_box-group*
- */NXcg_grid/cardinality-field*
- */NXcg_grid/cell_dimensions-field*
- */NXcg_grid/coordinate-field*
- */NXcg_grid/dimensionality-field*
- */NXcg_grid/extent-field*
- */NXcg_grid/identifier-field*
- */NXcg_grid/identifier_offset-field*
- */NXcg_grid/number_of_boundaries-field*
- */NXcg_grid/origin-field*
- */NXcg_grid/position-field*
- */NXcg_grid/symmetry-field*
- */NXcg_grid/TRANSFORMATIONS-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_grid.nxdl.xml

NXcg_half_edge_data_structure

Status:

base class, extends *NXobject*

Description:

Computational geeometry description of a half-edge data structure.

Such a data structure can be used to efficiently circulate around faces and iterate over vertices of a planar graph.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: The dimensionality, which has to be at least 2.

n_v: The number of vertices.

n_f: The number of faces.

n_he: The number of half-edges.

Groups cited:
none

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

number_of_vertices: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

number_of_faces: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

number_of_half_edges: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

vertex_identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

In this half-edge data structure vertex identifiers start at 1. Vertices are identified with consecutive integers up to number_of_vertices. This field can be used to document which constant integer has to be added to another set of vertex_identifier to assure that these other identifiers also start at 1.

face_identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

In this half-edge data structure face identifiers start at 1. Faces are identified with consecutive integers up to number_of_faces. This field can be used to document which constant integer has to be added to another set of face_identifier to assure that these other identifiers also start at 1.

The face identifier zero is reserved for the NULL face !

half_edge_identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

In this half-edge data structure half-edge identifiers start at 1. Half-edges are identified with consecutive integers up to number_of_half_edges. This field can be used to document which constant integer has to be added to another set of half_edge_identifier to assure that these other identifiers also start at 1.

position: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_v, d]) {units=*NX_LENGTH*}

The position of the vertices.

vertex_incident_half_edge: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_v]) {units=*NX_UNITLESS*}

Identifier of the incident half-edge.

face_half_edge: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_f]) {units=*NX_UNITLESS*}

Identifier of the (starting)/associated half-edge of the face.

half_edge_vertex_origin: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_he]) {units=*NX_UNITLESS*}

The identifier of the vertex from which this half-edge is outwards pointing.

half_edge_twin: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_he]) {units=*NX_UNITLESS*}

Identifier of the associated oppositely pointing half-edge.

half_edge_incident_face: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_he]) {units=*NX_UNITLESS*}

If the half-edge is a boundary half-edge the incident face identifier is NULL, i.e. 0.

half_edge_next: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_he]) {units=*NX_UNITLESS*}

Identifier of the next half-edge.

half_edge_prev: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_he]) {units=*NX_UNITLESS*}

Identifier of the previous half-edge.

weinberg_vector: (optional) *NX_CHAR*

Users are referred to the literature for the background of L. Weinberg's work about topological characterization of planar graphs:

- L. Weinberg 1966a,

- L. Weinberg, 1966b,
- E. A. Lazar et al.

and how this work can e.g. be applied in space-filling tessellations of microstructural objects like crystals/grains.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_half_edge_data_structure/dimensionality-field*
- */NXcg_half_edge_data_structure/face_half_edge-field*
- */NXcg_half_edge_data_structure/face_identifier_offset-field*
- */NXcg_half_edge_data_structure/half_edge_identifier_offset-field*
- */NXcg_half_edge_data_structure/half_edge_incident_face-field*
- */NXcg_half_edge_data_structure/half_edge_next-field*
- */NXcg_half_edge_data_structure/half_edge_prev-field*
- */NXcg_half_edge_data_structure/half_edge_twin-field*
- */NXcg_half_edge_data_structure/half_edge_vertex_origin-field*
- */NXcg_half_edge_data_structure/number_of_faces-field*
- */NXcg_half_edge_data_structure/number_of_half_edges-field*
- */NXcg_half_edge_data_structure/number_of_vertices-field*
- */NXcg_half_edge_data_structure/position-field*
- */NXcg_half_edge_data_structure/vertex_identifier_offset-field*
- */NXcg_half_edge_data_structure/vertex_incident_half_edge-field*
- */NXcg_half_edge_data_structure/weinberg_vector-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_half_edge_data_structure.nxdl.xml

NXcg_hexahedron_set

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a set of hexahedra in Euclidean space.

The hexahedra do not have to be connected, can have different size, can intersect, and be rotated. This class can also be used to describe cuboids or cubes, axis-aligned or not. The class represents different access and description levels to offer both applied scientists and computational geometry experts to use the same base class but rather their specific view on the data:

- Most simple many experimentalists wish to communicate dimensions/the size of specimens. In this case the alignment with axes is not relevant as eventually the only relevant information to convey is the volume of the specimen.
- In many cases, take for instance an experiment where a specimen was taken from a specifically deformed piece of material, e.g. cold-rolled, channel-die deformed, the orientation of the specimen edges with the experiment coordinate system can be of very high relevance. Examples include to know which specimen edge is parallel to the rolling, the transverse, or the normal direction.
- Sufficient to pinpoint the sample and laboratory/experiment coordinate system, the above-mentioned descriptions are not detailed enough though to create a CAD model of the specimen.
- Therefore, groups and fields for an additional, computational-geometry- based view of the hexahedra is offered which serve different computational tasks: storage-oriented simple views or detailed topological/graph-based descriptions.

Hexahedra are important geometrical primitives, which are among the most frequently used elements in finite element meshing/modeling.

Hexahedra have to be non-degenerated, closed, and built of polygons which are not self-intersecting.

The term hexahedra will be used throughout this base class but includes the especially in engineering and more commonly used special cases, cuboid, cube, box, axis-aligned bounding box (AABB), optimal bounding box (OBB).

An axis-aligned bounding box is a common data object in computational science and codes to represent a cuboid whose edges are aligned with a coordinate system. As a part of binary trees these are important data objects for time as well as space efficient queries of geometric primitives in techniques like kd-trees.

An optimal bounding box is a common data object which provides the best tight fitting box about an arbitrary object. In general such boxes are rotated. Exact and substantially faster in practice approximate algorithms exist for computing optimal or near optimal bounding boxes for point sets.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

c: The cardinality of the set, i.e. the number of hexahedra.

Groups cited:
NXcg_face_list_data_structure, *NXcg_half_edge_data_structure*, *NXcg_unit_normal_set*, *NXorientation_set*, *NXtransformations*

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Obligatory value: 3

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

shape: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_LENGTH*}

A qualitative description of each hexahedron/cuboid/cube/box.

length: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_LENGTH*}

Qualifier how one edge is longer than all other edges of the hexahedra. Often the term length is associated with one edge being parallel to an axis of the coordinate system.

width: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_LENGTH*}

Qualifier often used to describe the length of an edge within a specific coordinate system.

height: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_LENGTH*}

Qualifier often used to describe the length of an edge within a specific coordinate system.

center: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_LENGTH*}

Position of the geometric center, which often is but not necessarily has to be the center_of_mass of the hexahedrally-shaped sample/sample part.

volume: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_VOLUME*}

surface_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*}

Total area (of all six faces) of each hexahedron.

face_area: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 6]) {units=*NX_AREA*}

Area of each of the six faces of each hexahedron.

is_box: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [c])

Specifies if the hexahedra represent cuboids or cubes eventually rotated ones but at least not too exotic six-faced polyhedra.

is_axis_aligned: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [c])

Only to be used if is_box is present. In this case, this field describes whether hexahedra are boxes whose primary edges are parallel to the axes of the Cartesian coordinate system.

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing hexahedra. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Integer used to distinguish hexahedra for explicit indexing.

TRANSFORMATIONS: (optional) *NXtransformations*

Reference to or definition of a coordinate system with which the qualifiers and mesh data are interpretable.

orientation: (optional) *NXorientation_set*

vertex_normal: (optional) *NXcg_unit_normal_set*

edge_normal: (optional) *NXcg_unit_normal_set*

face_normal: (optional) *NXcg_unit_normal_set*

hexahedra: (optional) *NXcg_face_list_data_structure*

A simple approach to describe the entire set of hexahedra when the main intention is to store the shape of the hexahedra for visualization.

hexahedron: (optional) *NXcg_face_list_data_structure*

Disentangled representations of the mesh of specific hexahedra.

hexahedron_half_edge: (optional) *NXcg_half_edge_data_structure*

Disentangled representation of the planar graph that each hexahedron represents. Such a description simplifies topological processing or analyses of mesh primitive operations and neighborhood queries.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_hexahedron_set/cardinality-field*
- */NXcg_hexahedron_set/center-field*
- */NXcg_hexahedron_set/dimensionality-field*
- */NXcg_hexahedron_set/edge_normal-group*
- */NXcg_hexahedron_set/face_area-field*
- */NXcg_hexahedron_set/face_normal-group*
- */NXcg_hexahedron_set/height-field*
- */NXcg_hexahedron_set/hexahedra-group*
- */NXcg_hexahedron_set/hexahedron-group*
- */NXcg_hexahedron_set/hexahedron_half_edge-group*
- */NXcg_hexahedron_set/identifier-field*
- */NXcg_hexahedron_set/identifier_offset-field*
- */NXcg_hexahedron_set/is_axis_aligned-field*
- */NXcg_hexahedron_set/is_box-field*
- */NXcg_hexahedron_set/length-field*
- */NXcg_hexahedron_set/orientation-group*
- */NXcg_hexahedron_set/shape-field*
- */NXcg_hexahedron_set/surface_area-field*
- */NXcg_hexahedron_set/TRANSFORMATIONS-group*
- */NXcg_hexahedron_set/vertex_normal-group*
- */NXcg_hexahedron_set/volume-field*
- */NXcg_hexahedron_set/width-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_hexahedron_set.nxdl.xml

NXcg_marching_cubes

Status:

base class, extends *NXobject*

Description:

Computational geometry description of the marching cubes algorithm.

Documenting which specific version was used can help to understand how robust the results are with respect to the topology of the triangulation.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXcg_grid

Structure:

implementation: (optional) *NX_CHAR*

Reference to the specific implementation of marching cubes used.

See for example the following papers for details about how to identify a DOI which specifies the implementation used:

- W. E. Lorensen
- T. S. Newman and H. Yi

The value placed here should be a DOI. If there are no specific DOI or details write not_further_specified, or give at least a free-text description.

program: (optional) *NX_CHAR*

Commercial or otherwise given name to the program which was used.

@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.

grid: (optional) *NXcg_grid*

Reference/link to and/or details of the grid on which a specific marching cubes algorithm implementation is operating.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_marching_cubes/grid-group*
- */NXcg_marching_cubes/implementation-field*
- */NXcg_marching_cubes/program-field*
- */NXcg_marching_cubes/program@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_marching_cubes.nxdl.xml

NXcg_parallelogram_set

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a set of parallelograms in Euclidean space.

The parallelograms do not have to be connected, can have different size, can intersect, and be rotated. This class can also be used to describe rectangles or squares, axis-aligned or not. The class represents different access and description levels to offer both applied scientists and computational geometry experts to use the same base class but rather their specific view on the data:

- Most simple many experimentalists wish to communicate dimensions/the size of e.g. a region of interest in the 2D plane. In this case the alignment with axes is not relevant as eventually relevant is only the area of the ROI.
- In other cases the extent of the parallelogram is relevant though.
- Finally in CAD models we would like to specify the polygon which is parallelogram represents.

Parallelograms are important geometrical primitives. Not so much because of their uses in nowadays, thanks to improvements in computing power, not so frequently any longer performed 2D simulation. Many scanning experiments probe though parallelogram-shaped ROIs on the surface of samples.

Parallelograms have to be non-degenerated, closed, and built of polygons which are not self-intersecting.

The term parallelogram will be used throughout this base class but includes the especially in engineering and more commonly used special cases, rectangle, square, 2D box, axis-aligned bounding box (AABB), or optimal bounding box (OBB) but here the analogous 2D cases.

An axis-aligned bounding box is a common data object in computational science and codes to represent a rectangle whose edges are aligned with the axes of a coordinate system. As a part of binary trees these are important data objects for time- as well as space-efficient queries of geometric primitives in techniques like kd-trees.

An optimal bounding box is a common data object which provides the best tight fitting box about an arbitrary object. In general such boxes are rotated. Other than in 3D dimensions the rotation calipher method offers a rigorous approach to compute optimal bounding boxes in 2D.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

c: The cardinality of the set, i.e. the number of parallelograms.

Groups cited:
NXcg_face_list_data_structure, *NXcg_unit_normal_set*, *NXorientation_set*, *NXtransformations*

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Obligatory value: 2

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

shape: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 2]) {units=*NX_LENGTH*}

A qualitative description of each parallelogram/rectangle/square/box.

length: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_LENGTH*}

Qualifier how one edge is longer than all the other edge of the parallelogam. Often the term length is associated with one edge being parallel to an axis of the coordinate system.

width: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_LENGTH*}

Qualifier often used to describe the length of an edge within a specific coordinate system.

center: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 2]) {units=*NX_LENGTH*}

Position of the geometric center, which often is but not necessarily has to be the center_of_mass of the parallelogram.

surface_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*}

is_axis_aligned: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [c])

Only to be used if is_box is present. In this case, this field describes whether parallelograms are rectangles/squares whose primary edges are parallel to the axes of the Cartesian coordinate system.

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing parallelograms. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Integer used to distinguish parallelograms for explicit indexing.

TRANSFORMATIONS: (optional) *NXtransformations*

Reference to or definition of a coordinate system with which the qualifiers and mesh data are interpretable.

orientation: (optional) *NXorientation_set*

vertex_normal: (optional) *NXcg_unit_normal_set*

edge_normal: (optional) *NXcg_unit_normal_set*

face_normal: (optional) *NXcg_unit_normal_set*

parallelograms: (optional) *NXcg_face_list_data_structure*

A simple approach to describe the entire set of parallelograms when the main intention is to store the shape of the parallelograms for visualization.

parallelogram: (optional) *NXcg_face_list_data_structure*

Disentangled representations of the mesh of specific parallelograms.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_parallelogram_set/cardinality-field*
- */NXcg_parallelogram_set/center-field*
- */NXcg_parallelogram_set/dimensionality-field*
- */NXcg_parallelogram_set/edge_normal-group*
- */NXcg_parallelogram_set/face_normal-group*
- */NXcg_parallelogram_set/identifier-field*
- */NXcg_parallelogram_set/identifier_offset-field*
- */NXcg_parallelogram_set/is_axis_aligned-field*
- */NXcg_parallelogram_set/length-field*
- */NXcg_parallelogram_set/orientation-group*
- */NXcg_parallelogram_set/parallelogram-group*
- */NXcg_parallelogram_set/parallelograms-group*
- */NXcg_parallelogram_set/shape-field*
- */NXcg_parallelogram_set/surface_area-field*
- */NXcg_parallelogram_set/TRANSFORMATIONS-group*
- */NXcg_parallelogram_set/vertex_normal-group*
- */NXcg_parallelogram_set/width-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_parallelogram_set.nxdl.xml

NXcg_point_set

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a set of points in Euclidean space.

The relevant coordinate system should be referred to in the NXtransformations instance. Points may have an associated time value; however users are advised to store time data of point sets rather as instances of time events, where for each point in time there is an NXcg_point_set instance which specifies the points locations. This is a frequent situation in experiments and computer simulations, where positions of points are taken at the same point in time; and therefore an additional time array would demand to store redundant pieces of information for each point.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: The dimensionality, which has to be at least 1.

c: The cardinality of the set, i.e. the number of points.

Groups cited:
NXtransformations

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing identifiers for points. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Integer used to distinguish points for explicit indexing.

position: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, d]) {units=*NX_LENGTH*}

The array of point coordinates.

time: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_TIME*}

The optional array of time for each vertex.

TRANSFORMATIONS: (optional) *NXtransformations*

Reference to or definition of a coordinate system with which the positions and directions are interpretable.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_point_set/cardinality-field*
- */NXcg_point_set/dimensionality-field*
- */NXcg_point_set/identifier-field*
- */NXcg_point_set/identifier_offset-field*
- */NXcg_point_set/position-field*
- */NXcg_point_set/time-field*
- */NXcg_point_set/TRANSFORMATIONS-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_point_set.nxdl.xml

NXcg_polygon_set

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a set of polygons in Euclidean space.

Polygons are related are specialized polylines:

- A polygon is a geometric primitive that is bounded by a closed polyline
- All vertices of this polyline lay in the d-1 dimensional plane. whereas vertices of a polyline do not necessarily lay on a plane.
- A polygon has at least three vertices.

Each polygon is built from a sequence of vertices (points with identifiers). The members of a set of polygons may have a different number of vertices. Sometimes a collection/set of polygons is referred to as a soup of polygons.

As three-dimensional objects, a set of polygons can be used to define the hull of what is effectively a polyhedron; however users are advised to use the specific NXcg_polyhedron_set base class if they wish to describe closed polyhedra. Even more general complexes can be thought, for instance piecewise-linear complexes, as these can have holes though, polyhedra without holes are one subclass of such complexes, users should rather design an own base class e.g. NXcg_polytope_set to describe such even more complex primitives.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: The dimensionality, which has to be either 2 or 3.

c: The cardinality of the set, i.e. the number of polygons.

n_total: The total number of vertices when visiting every polygon.

Groups cited:

NXcg_face_list_data_structure, *NXcg_hexahedron_set*, *NXcg_unit_normal_set*

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Any of these values: `2` | `3`

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

number_of_total_vertices: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing polygons. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Integer used to distinguish polygons for explicit indexing.

area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*}

edge_length: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_LENGTH*}

The accumulated length of the polygon edge.

interior_angle: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_total]) {units=*NX_ANGLE*}

Array of interior angles. There are many values per polygon as number_of_vertices. The angle is the angle at the specific vertex, i.e. between the adjoining edges of the vertex according to the sequence in the polygons array. Usually, the winding_order field is required to interpret the value.

shape: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Curvature type:

- 0 - unspecified,
- 1 - convex,
- 2 - concave

center: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, d]) {units=*NX_LENGTH*}

The center of mass of each polygon.

polygons: (optional) *NXcg_face_list_data_structure*

A simple approach to describe the entire set of polygons when the main intention is to store the shape of the polygons for visualization.

vertex_normal: (optional) *NXcg_unit_normal_set*

edge_normal: (optional) *NXcg_unit_normal_set*

face_normal: (optional) *NXcg_unit_normal_set*

bounding_box: (optional) *NXcg_hexahedron_set*

Axis-aligned or (approximate) (optimal) bounding boxes to each polygon.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_polygon_set/area-field*
- */NXcg_polygon_set/bounding_box-group*
- */NXcg_polygon_set/cardinality-field*
- */NXcg_polygon_set/center-field*
- */NXcg_polygon_set/dimensionality-field*
- */NXcg_polygon_set/edge_length-field*
- */NXcg_polygon_set/edge_normal-group*
- */NXcg_polygon_set/face_normal-group*
- */NXcg_polygon_set/identifier-field*
- */NXcg_polygon_set/identifier_offset-field*

- */NXcg_polygon_set/interior_angle-field*
- */NXcg_polygon_set/number_of_total_vertices-field*
- */NXcg_polygon_set/polygons-group*
- */NXcg_polygon_set/shape-field*
- */NXcg_polygon_set/vertex_normal-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_polygon_set.nxdl.xml

NXcg_polyhedron_set

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a polyhedra in Euclidean space.

Polyhedra, also so-called cells (especially in the convex of tessellations), here described have to be all non-degenerated, closed, built of and thus built out of not-self-intersecting polygon meshes. Polyhedra may make contact, so that this base class can be used for a future description of tessellations.

For more complicated manifolds and especially for polyhedra with holes, users are advised to check if their particular needs are described by creating (eventually customized) instances of an NXcg_polygon_set, which can be extended for the description of piecewise-linear complexes.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

c: The cardinality of the set, i.e. the number of polyhedra.

n_e_total: The total number of edges for all polyhedra.

n_f_total: The total number of faces for all polyhedra.

Groups cited:
NXcg_face_list_data_structure, *NXcg_half_edge_data_structure*, *NXcg_unit_normal_set*, *NXtransformations*

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Obligatory value: 3

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

volume: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_VOLUME*}

Interior volume

center: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_LENGTH*}

Position of the geometric center, which often is but not necessarily has to be the center_of_mass of the polyhedra.

surface_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*}

Total surface area as the sum of all faces.

number_of_faces: (optional) *NX_POSINT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

The number of faces for each polyhedron. Faces of adjoining polyhedra are counted for each polyhedron. This field can be used to interpret the array/field with the individual area values for each face.

face_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_f_total]) {units=*NX_AREA*}

Area of each of the four triangular faces of each tetrahedron.

number_of_edges: (optional) *NX_POSINT*

The number of edges for each polyhedron. Edges of adjoining polyhedra are counterd for each polyhedron. This field can be used to interpret the array/field with the individual length for each edge.

edge_length: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_e_total]) {units=*NX_LENGTH*}

Length of each edge of each tetrahedron.

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing polyhedra. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Integer used to distinguish polyhedra for explicit indexing.

TRANSFORMATIONS: (optional) *NXtransformations*

Reference to or definition of a coordinate system with which the qualifiers and mesh data are interpretable.

vertex_normal: (optional) *NXcg_unit_normal_set*

edge_normal: (optional) *NXcg_unit_normal_set*

face_normal: (optional) *NXcg_unit_normal_set*

polyhedra: (optional) *NXcg_face_list_data_structure*

A simple approach to describe the entire set of polyhedra when the main intention is to store the shape of the polyhedra for visualization.

polyhedron: (optional) *NXcg_face_list_data_structure*

Disentangled representations of the mesh of specific polyhedron.

polyhedron_half_edge: (optional) *NXcg_half_edge_data_structure*

Disentangled representation of the planar graph that each polyhedron represents. Such a description simplifies topological processing or analyses of mesh primitive operations and neighborhood queries.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_polyhedron_set/cardinality-field*
- */NXcg_polyhedron_set/center-field*
- */NXcg_polyhedron_set/dimensionality-field*
- */NXcg_polyhedron_set/edge_length-field*
- */NXcg_polyhedron_set/edge_normal-group*
- */NXcg_polyhedron_set/face_area-field*
- */NXcg_polyhedron_set/face_normal-group*
- */NXcg_polyhedron_set/identifier-field*
- */NXcg_polyhedron_set/identifier_offset-field*
- */NXcg_polyhedron_set/number_of_edges-field*
- */NXcg_polyhedron_set/number_of_faces-field*
- */NXcg_polyhedron_set/polyhedra-group*
- */NXcg_polyhedron_set/polyhedron-group*
- */NXcg_polyhedron_set/polyhedron_half_edge-group*
- */NXcg_polyhedron_set/surface_area-field*
- */NXcg_polyhedron_set/TRANSFORMATIONS-group*
- */NXcg_polyhedron_set/vertex_normal-group*
- */NXcg_polyhedron_set/volume-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_polyhedron_set.nxdl.xml

NXcg_polyline_set

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a set of polylines in Euclidean space.

Each polyline is built from a sequence of vertices (points with identifiers). Each polyline must have a start and an end point. The sequence describes the positive traversal along the polyline when walking from the start vertex to the end/last vertex.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: The dimensionality, which has to be at least 1.

c: The cardinality of the set, i.e. the number of polylines.

n_v: The number of vertices, supporting the polylines.

n_total: The total number of vertices traversed when visiting every polyline.

Groups cited:
NXcg_unit_normal_set, *NXtransformations*

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

number_of_total_vertices: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

The total number of vertices, irrespective of their eventual uniqueness, i.e. the total number of vertices that have to be visited when walking along each polyline.

number_of_vertices: (optional) *NX_POSINT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Array which specifies of how many vertices each polyline is built. The number of vertices represent the total number of vertices for each polyline, irrespectively whether vertices are shared or not. See the docstring for polylines for further details about how a set with different polyline members should be stored.

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing polylines. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Integer used to distinguish polylines for explicit indexing.

vertices: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_v, d]) {units=*NX_LENGTH*}

Positions of the vertices which support the members of the polyline set.

Users are encouraged to reduce the vertices to unique set of positions and vertices as this supports a more efficient storage of the geometry data. It is also possible though to store the vertex positions naively in which case vertices_are_unique is likely False. Naively here means that one for example stores each vertex of a triangle mesh even though many vertices are shared between triangles and thus the positions of these vertices do not have to be duplicated.

vertices_are_unique: (optional) *NX_BOOLEAN*

If true indicates that the vertices are all placed at different positions and have different identifiers, i.e. no points overlap or are counted twice.

polylines: (optional) *NX_INT* (Rank: 1, Dimensions: [n_total]) {units=*NX_UNITLESS*}

Sequence of vertex identifiers which describe each polyline.

A trivial example is a set with two polylines with three vertices each. If the polylines meet in a junction, say the second vertex is shared and marking the junction between the two polylines, it is possible that there are only five unique positions suggesting five unique vertices.

A non-trivial example is a set with several polylines, each of which with eventually different number of vertices. The array stores the vertex identifiers in the order how the polylines are visited:

The first entry is the identifier of the start vertex of the first polyline, followed by the second vertex of the first polyline, until the last vertex of the polyline. Thereafter, the start vertex of the second polyline, and so on and so forth. Using the (cumulated) counts in number_of_vertices, the vertices of the n-th polyline can be accessed on the following array index interval: $[\sum_{i=0}^{i=N-1}, \sum_{i=0}^{i=N}]$.

length: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_LENGTH*}

The length of each polyline.

is_closed: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [c])

If true specifies that a polyline is closed, i.e. its end point is connected to the start point.

TRANSFORMATIONS: (optional) *NXtransformations*

Reference to or definition of a coordinate system with which the qualifiers and polyline data are interpretable.

vertex_normal: (optional) *NXcg_unit_normal_set*

edge_normal: (optional) *NXcg_unit_normal_set*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_polyline_set/cardinality-field*
- */NXcg_polyline_set/dimensionality-field*
- */NXcg_polyline_set/edge_normal-group*
- */NXcg_polyline_set/identifier-field*
- */NXcg_polyline_set/identifier_offset-field*
- */NXcg_polyline_set/is_closed-field*
- */NXcg_polyline_set/length-field*
- */NXcg_polyline_set/number_of_total_vertices-field*
- */NXcg_polyline_set/number_of_vertices-field*
- */NXcg_polyline_set/polylines-field*
- */NXcg_polyline_set/TRANSFORMATIONS-group*
- */NXcg_polyline_set/vertex_normal-group*
- */NXcg_polyline_set/vertices-field*
- */NXcg_polyline_set/vertices_are_unique-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_polyline_set.nxdl.xml

NXcg_roi_set

Status:

base class, extends *NXobject*

Description:

Base class to hold geometric primitives.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXcg_cylinder_set, *NXcg_ellipsoid_set*, *NXcg_polyhedron_set*, *NXcg_sphere_set*

Structure:

CG_SPHERE_SET: (optional) *NXcg_sphere_set*

CG_ELLIPSOID_SET: (optional) *NXcg_ellipsoid_set*

CG_CYLINDER_SET: (optional) *NXcg_cylinder_set*

CG_POLYHEDRON_SET: (optional) *NXcg_polyhedron_set*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_roi_set/CG_CYLINDER_SET-group*
- */NXcg_roi_set/CG_ELLIPSOID_SET-group*
- */NXcg_roi_set/CG_POLYHEDRON_SET-group*
- */NXcg_roi_set/CG_SPHERE_SET-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_roi_set.nxdl.xml

NXcg_sphere_set

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a set of spheres in Euclidean space.

Each sphere can have a different radius.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: The dimensionality, which has to be at least 2.

c: The cardinality of the set, i.e. the number of circles or spheres.

Groups cited:
NXtransformations

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing identifiers for spheres. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Integer used to distinguish spheres for explicit indexing.

center: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, d]) {units=*NX_LENGTH*}

Geometric center of the spheres. This can be the center of mass. Dimensionality and cardinality of the point and sphere set have to match. The identifier_offset and identifier field of NXcg_point_set do not need to be used as they should be same as the identifier_offset and the identifier for the spheres.

radius: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

In the case that all spheres have the same radius.

radii: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_LENGTH*}

In the case that spheres have different radius use this instead of the radius field.

is_closed: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [c])

Are the spheres closed or hollow?

volume: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_ANY*}

surface_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_ANY*}

TRANSFORMATIONS: (optional) *NXtransformations*

Reference to or definition of a coordinate system with which the positions and directions are interpretable.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_sphere_set/cardinality-field*
- */NXcg_sphere_set/center-field*
- */NXcg_sphere_set/dimensionality-field*
- */NXcg_sphere_set/identifier-field*
- */NXcg_sphere_set/identifier_offset-field*
- */NXcg_sphere_set/is_closed-field*
- */NXcg_sphere_set/radii-field*

- */NXcg_sphere_set/radius-field*
- */NXcg_sphere_set/surface_area-field*
- */NXcg_sphere_set/TRANSFORMATIONS-group*
- */NXcg_sphere_set/volume-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_sphere_set.nxdl.xml

NXcg_tetrahedron_set

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a set of tetrahedra in Euclidean space.

The tetrahedra do not have to be connected. As tetrahedral elements they are among hexahedral elements one of the most frequently used geometric primitive for meshing and describing volumetric and surface descriptions of objects at the continuum scale.

A set of tetrahedra in 3D Euclidean space.

The tetrahedra do not have to be connected, can have different size, can intersect, and be rotated.

Tetrahedra are the simplest and thus important geometrical primitive. They are frequently used as elements in finite element meshing/modeling.

Tetrahedra have to be non-degenerated, closed, and built of triangles which are not self-intersecting.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

c: The cardinality of the set, i.e. the number of tetrahedra.

Groups cited:
NXcg_face_list_data_structure, *NXcg_half_edge_data_structure*, *NXcg_unit_normal_set*, *NXtransformations*

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Obligatory value: 3

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

volume: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_VOLUME*}

Interior volume

center: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_LENGTH*}

Position of the geometric center, which often is but not necessarily has to be the center_of_mass of the tetrahedra.

surface_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*}

Total surface area as the sum of all four triangular faces.

face_area: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 4]) {units=*NX_AREA*}

Area of each of the four triangular faces of each tetrahedron.

edge_length: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 6]) {units=*NX_LENGTH*}

Length of each edge of each tetrahedron.

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing tetrahedra. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Integer used to distinguish tetrahedra for explicit indexing.

TRANSFORMATIONS: (optional) *NXtransformations*

Reference to or definition of a coordinate system with which the qualifiers and mesh data are interpretable.

vertex_normal: (optional) *NXcg_unit_normal_set*

edge_normal: (optional) *NXcg_unit_normal_set*

face_normal: (optional) *NXcg_unit_normal_set*

tetrahedra: (optional) *NXcg_face_list_data_structure*

A simple approach to describe the entire set of tetrahedra when the main intention is to store the shape of the tetrahedra for visualization. should take the possibility to describe

tetrahedron: (optional) *NXcg_face_list_data_structure*

Disentangled representations of the mesh of specific tetrahedra.

tetrahedron_half_edge: (optional) *NXcg_half_edge_data_structure*

Disentangled representation of the planar graph that each tetrahedron represents. Such a description simplifies topological processing or analyses of mesh primitive operations and neighborhood queries.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_tetrahedron_set/cardinality-field*
- */NXcg_tetrahedron_set/center-field*
- */NXcg_tetrahedron_set/dimensionality-field*
- */NXcg_tetrahedron_set/edge_length-field*
- */NXcg_tetrahedron_set/edge_normal-group*
- */NXcg_tetrahedron_set/face_area-field*
- */NXcg_tetrahedron_set/face_normal-group*
- */NXcg_tetrahedron_set/identifier-field*
- */NXcg_tetrahedron_set/identifier_offset-field*

- */NXcg_tetrahedron_set/surface_area-field*
- */NXcg_tetrahedron_set/tetrahedra-group*
- */NXcg_tetrahedron_set/tetrahedron-group*
- */NXcg_tetrahedron_set/tetrahedron_half_edge-group*
- */NXcg_tetrahedron_set/TRANSFORMATIONS-group*
- */NXcg_tetrahedron_set/vertex_normal-group*
- */NXcg_tetrahedron_set/volume-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_tetrahedron_set.nxdl.xml

NXcg_triangle_set

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a set of triangles in Euclidean space.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: The dimensionality, which has to be at least 2.

c: The cardinality of the set, i.e. the number of triangles.

n_unique: The number of unique vertices supporting the triangles.

Groups cited:
NXcg_face_list_data_structure, *NXcg_hexahedron_set*, *NXcg_unit_normal_set*, *NXtransformations*

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

number_of_unique_vertices: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing triangles. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Integer used to distinguish triangles for explicit indexing.

area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*}

edge_length: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_LENGTH*}

Array of edge length values. For each triangle the edge length is reported for the edges traversed according to the sequence in which vertices are indexed in triangles.

interior_angle: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_ANGLE*}

Array of interior angle values. For each triangle the angle is reported for the angle opposite to the edges which are traversed according to the sequence in which vertices are indexed in triangles.

center: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, d]) {units=*NX_LENGTH*}

The center of mass of each polygon.

TRANSFORMATIONS: (optional) *NXtransformations*

Reference to or definition of a coordinate system with which the qualifiers and primitive data are interpretable.

triangles: (optional) *NXcg_face_list_data_structure*

A simple approach to describe the entire set of triangles when the main intention is to store the shape of the triangles for visualization.

vertex_normal: (optional) *NXcg_unit_normal_set*

edge_normal: (optional) *NXcg_unit_normal_set*

face_normal: (optional) *NXcg_unit_normal_set*

bounding_box: (optional) *NXcg_hexahedron_set*

Axis-aligned or (approximate) (optimal) bounding boxes to each polygon.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_triangle_set/area-field*
- */NXcg_triangle_set/bounding_box-group*
- */NXcg_triangle_set/cardinality-field*
- */NXcg_triangle_set/center-field*
- */NXcg_triangle_set/dimensionality-field*
- */NXcg_triangle_set/edge_length-field*
- */NXcg_triangle_set/edge_normal-group*
- */NXcg_triangle_set/face_normal-group*
- */NXcg_triangle_set/identifier-field*
- */NXcg_triangle_set/identifier_offset-field*
- */NXcg_triangle_set/interior_angle-field*
- */NXcg_triangle_set/number_of_unique_vertices-field*
- */NXcg_triangle_set/TRANSFORMATIONS-group*
- */NXcg_triangle_set/triangles-group*
- */NXcg_triangle_set/vertex_normal-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_triangle_set.nxdl.xml

NXcg_triangulated_surface_mesh

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a mesh of triangles.

The mesh may be self-intersecting and have holes but the triangles must not be degenerated.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXcg_half_edge_data_structure, *NXcg_triangle_set*

Structure:

CG_TRIANGLE_SET: (optional) *NXcg_triangle_set*

CG_HALF_EDGE_DATA_STRUCTURE: (optional) *NXcg_half_edge_data_structure*

A graph-based approach to describe the mesh when it is also desired to perform topological processing or analyses on the mesh.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_triangulated_surface_mesh/CG_HALF_EDGE_DATA_STRUCTURE-group*
- */NXcg_triangulated_surface_mesh/CG_TRIANGLE_SET-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_triangulated_surface_mesh.nxdl.xml

NXcg_unit_normal_set

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a set of (oriented) unit normal vectors.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: The dimensionality, which has to be at least 2.

c: The cardinality of the set, i.e. the number of unit normals.

Groups cited:
none

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

normals: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [c, d]) {units=*NX_LENGTH*}

Direction of each normal

orientation: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Qualifier how which specifically oriented normal to its primitive each normal represents.

- 0 - undefined
- 1 - outer
- 2 - inner

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcg_unit_normal_set/cardinality-field*
- */NXcg_unit_normal_set/dimensionality-field*
- */NXcg_unit_normal_set/normals-field*
- */NXcg_unit_normal_set/orientation-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcg_unit_normal_set.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:
NXfabrication

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.

FABRICATION: (optional) *NXfabrication*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXchamber/description-field*
- */NXchamber/FABRICATION-group*
- */NXchamber/name-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXchamber.nxdl.xml

NXchemical_composition

Status:

base class, extends *NXobject*

Description:

(Chemical) composition of a sample or a set of things.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n: The number of samples or things.

Groups cited:
NXion

Structure:

total: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

Total based on which composition information is normalized.

ION: (optional) *NXion*

count: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

Count or weight which, when divided by total yields the composition of this element, isotope, molecule or ion.

composition: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n]) {units=*NX_DIMENSIONLESS*}

Count divided by total in atom percent.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXchemical_composition/ION-group*
- */NXchemical_composition/ION/composition-field*
- */NXchemical_composition/ION/count-field*
- */NXchemical_composition/total-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXchemical_composition.nxdl.xml

NXcircuit_board

Status:

base class, extends *NXobject*

Description:

Circuit board with e.g. ADC and/or DAC electronic components.

Currently used to store the settings of the so-called magboards used in Nion electron microscopes but likely this could be a useful base class for substantially more use cases where details at a deep technical instrument design level are relevant or important.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXadc, *NXdac*

Structure:

relay: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

TBD by Nion Co.

DAC: (optional) *NXdac*

ADC: (optional) *NXadc*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcircuit_board/ADC-group*
- */NXcircuit_board/DAC-group*
- */NXcircuit_board/relay-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcircuit_board.nxdl.xml

NXclustering

Status:

base class, extends *NXobject*

Description:

Metadata to the results of a clustering analysis.

Clustering algorithms are routine tools to segment a set of objects/primitives into groups, objects of different type. A plethora of algorithms have been proposed for geometric primitives as objects, such as points, triangles, or (abstract) objects.

This base class considers metadata and results of one clustering applied to a set in which objects are either categorized as noise or belonging to a cluster, specifically here only one cluster.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_lbl_num: Number of numeral labels per object.

n_lbl_cat: Number of categorical labels per object.

n_cluster: Total number of clusters detected.

Groups cited:
none

Structure:

number_of_numeric_labels: (optional) *NX_UINT* {units=*NX_UNITLESS*}

How many numeric labels does each object have.

number_of_categorical_labels: (optional) *NX_UINT* {units=*NX_UNITLESS*}

How many categorical labels does each object have.

objects: (optional) *NX_CHAR*

Reference to a set of objects investigated in a cluster analysis. Objects must have clear integer identifier.

numeric_label: (optional) *NX_NUMBER*

Reference to numeric attribute data for each object.

categorical_label: (optional) *NX_CHAR*

Reference to categorical attribute data for each object.

identifier_offset: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Which identifier is the first to be used to label a cluster.

The value should be chosen in such a way that special values can be resolved: * identifier_offset-1 indicates an object belongs to no cluster. * identifier_offset-2 indicates an object belongs to the noise category. Setting for instance identifier_offset to 1 recovers the commonly used case that objects of the noise category get values to -1 and unassigned points to 0.

unassigned: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Total number of objects categorized as unassigned.

noise: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Total number of objects categorized as noise.

number_of_cluster: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Total number of clusters (excluding noise and unassigned).

size: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_cluster]) {units=*NX_UNITLESS*}

Number of objects associated to each cluster. The labels are implicit, meaning the zeroth/first entry in the array belongs to the first cluster, the second entry to the second cluster and so on and so forth. The first cluster has the value of identifier_offset as its identifier. The second cluster has identifier_offset + 1, and so on and so forth.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXclustering/categorical_label-field*
- */NXclustering/identifier_offset-field*
- */NXclustering/noise-field*
- */NXclustering/number_of_categorical_labels-field*
- */NXclustering/number_of_cluster-field*
- */NXclustering/number_of_numeric_labels-field*
- */NXclustering/numeric_label-field*
- */NXclustering/objects-field*
- */NXclustering/size-field*
- */NXclustering/unassigned-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXclustering.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 collectioncolumn 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:

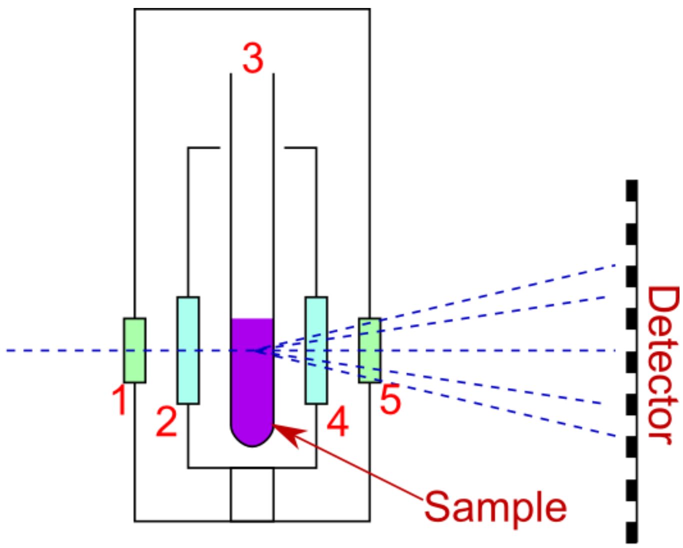


Fig. 12: A hypothetical capillary furnace. The beam passes from left to right (blue dashes), passing through window 1, then window 2, before passing through the downstream wall of the capillary. It is then scattered by the sample with scattered beams passing through the upstream wall of the capillary, 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.

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:

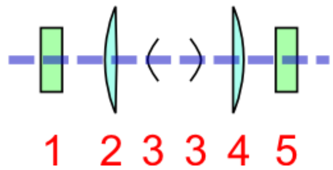


Fig. 13: A complete description of the shapes of the container elements with their orientation relative to the beam and also information on whether they are upstream or downstream of the sample is also therefore important. For example, although the windows 2 and 4 have the same shape, the path taken 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.

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 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.

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 technology partners use different naming schemes and models for quantifying the aberration coefficients.

The corrector in an electron microscope is composed of multiple lenses and multipole stigmators with vendor-specific details which are often undisclosed.

Symbols:

No symbol table

Groups cited:
NXaberration_model_ceos, *NXaberration_model_nion*, *NXfabrication*, *NXlens_em*, *NXprocess*, *NXtransformations*

Structure:

applied: (optional) *NX_BOOLEAN*

Was the corrector used?

name: (optional) *NX_CHAR*

Given name/alias.

description: (optional) *NX_CHAR*

Ideally, a (globally) unique persistent identifier, link, or text to a resource which gives further details. If this does not exist a free-text field to report further details about the corrector.

FABRICATION: (optional) *NXfabrication*

ZEMLIN_TABLEAU: (optional) *NXprocess*

Specific information about the concrete alignment procedure which is a process during which the corrector is configured to enable a calibrated usage of the microscope. **description**: (optional) *NX_CHAR*

Discouraged free-text field to add further details about the alignment procedure.

tilt_angle: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

The outer tilt angle of the beam in tableau aquisition.

exposure_time: (optional) *NX_FLOAT* {units=*NX_TIME*}

The exposure time of the single tilt images.

magnification: (optional) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

The factor of enlargement of the apparent size, not physical size, of an object.

PROCESS: (optional) *NXprocess*

Place for storing measured or estimated aberrations (for each image or final).

ceos: (optional) *NXaberration_model_ceos*

nion: (optional) *NXaberration_model_nion*

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/applied-field*
- */NXcorrector_cs/description-field*
- */NXcorrector_cs/FABRICATION-group*
- */NXcorrector_cs/LENS_EM-group*
- */NXcorrector_cs/name-field*
- */NXcorrector_cs/TRANSFORMATIONS-group*
- */NXcorrector_cs/ZEMLIN_TABLEAU-group*
- */NXcorrector_cs/ZEMLIN_TABLEAU/description-field*
- */NXcorrector_cs/ZEMLIN_TABLEAU/exposure_time-field*
- */NXcorrector_cs/ZEMLIN_TABLEAU/magnification-field*
- */NXcorrector_cs/ZEMLIN_TABLEAU/PROCESS-group*
- */NXcorrector_cs/ZEMLIN_TABLEAU/PROCESS/ceos-group*
- */NXcorrector_cs/ZEMLIN_TABLEAU/PROCESS/nion-group*
- */NXcorrector_cs/ZEMLIN_TABLEAU/tilt_angle-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcorrector_cs.nxdl.xml

NXcs_computer

Status:

base class, extends *NXobject*

Description:

Computer science description of a set of computing nodes.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXcs_cpu, *NXcs_gpu*, *NXcs_io_sys*, *NXcs_mm_sys*

Structure:

name: (optional) *NX_CHAR*

Given name/alias to the computing system, e.g. MyDesktop.

operating_system: (optional) *NX_CHAR*

Name of the operating system, e.g. Windows, Linux, Mac, Android.

@version: (optional) *NX_CHAR*

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.

uuid: (optional) *NX_CHAR*

Ideally a (globally) unique persistent identifier of the computer, i.e. the Universally Unique Identifier (UUID) of the computing node.

CS_CPU: (optional) *NXcs_cpu*

A list of physical processing units (can be multi-core chips).

CS_GPU: (optional) *NXcs_gpu*

A list of physical coprocessor/graphic cards/accelerator units.

CS_MM_SYS: (optional) *NXcs_mm_sys*

Details about the memory sub-system.

CS_IO_SYS: (optional) *NXcs_io_sys*

Details about the I/O sub-system.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcs_computer/CS_CPU-group*
- */NXcs_computer/CS_GPU-group*
- */NXcs_computer/CS_IO_SYS-group*
- */NXcs_computer/CS_MM_SYS-group*
- */NXcs_computer/name-field*
- */NXcs_computer/operating_system-field*
- */NXcs_computer/operating_system@version-attribute*
- */NXcs_computer/uuid-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcs_computer.nxdl.xml

NXcs_cpu

Status:

base class, extends *NXobject*

Description:

Computer science description of a central processing unit (CPU) of a computer.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXfabrication

Structure:

name: (optional) *NX_CHAR*

Given name of the CPU. Users should be as specific as possible.

FABRICATION: (optional) *NXfabrication*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcs_cpu/FABRICATION-group*
- */NXcs_cpu/name-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcs_cpu.nxdl.xml

NXcs_filter_boolean_mask

Status:

base class, extends *NXobject*

Description:

Computer science base class for packing and unpacking booleans.

One use case is processing of object sets (like point cloud data). When one applies e.g. a spatial filter to a set of points to define which points are analyzed and which not, it is useful to document which points were taken. One can store this information in a compact manner with an array of boolean values. If the value is True the point is taken, else it is not.

If the points are identified by an array of integer identifiers and an arbitrary spatial filtering, the boolean array will be filled with True and False values in an arbitrary manner. Especially when the number of points is large, for instance several thousands and more, some situations can be more efficiently stored if one would not store the boolean array but just list the identifiers of the points taken. For instance if within a set of 1000 points only one point is taken, it would take (naively) 4000 bits to store the array but only 32 bits to store e.g. the ID of that taken point. Of course the 4000 bit field is so sparse that it could be compressed resulting also in a substantial reduction of the storage demands. Therefore boolean masks are useful compact descriptions to store information about set memberships in a compact manner. In general it is true, though, that which representation is best, i.e. most compact (especially when compressed) depends strongly on occupation of the array.

This base class just bookkeeps metadata to inform software about necessary modulo operations to decode the set membership of each object. This is useful because the number of objects not necessarily is an integer multiple of the bit depth.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_objs: Number of entries (e.g. number of points or objects).

bitdepth: Number of bits assumed for the container datatype used.

n_total: Length of mask considering the eventual need for padding.

Groups cited:
none

Structure:

number_of_objects: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of objects represented by the mask.

bitdepth: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of bits assumed matching on a default datatype. (e.g. 8 bits for a C-style uint8).

mask: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_total]) {units=*NX_UNITLESS*}

The unsigned integer array representing the content of the mask. If padding is used the padded bits have to be set to 0.

identifier: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_object])

Link to/or array of identifiers for the objects. The decoded mask is interpreted consecutively, i.e. the first bit in the mask matches to the first identifier, the second bit in the mask to the second identifier and so on and so forth.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcs_filter_boolean_mask/bitdepth-field*
- */NXcs_filter_boolean_mask/identifier-field*
- */NXcs_filter_boolean_mask/mask-field*
- */NXcs_filter_boolean_mask/number_of_objects-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcs_filter_boolean_mask.nxdl.xml

NXcs_gpu

Status:

base class, extends *NXobject*

Description:

Computer science description of a graphic processing unit (GPU) of a computer.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXfabrication

Structure:

name: (optional) *NX_CHAR*

Given name of the GPU. Users should be as specific as possible.

FABRICATION: (optional) *NXfabrication*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcs_gpu/FABRICATION-group*
- */NXcs_gpu/name-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcs_gpu.nxdl.xml

NXcs_io_obj

Status:

base class, extends *NXobject*

Description:

Computer science description of a storage object in an input/output system.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXfabrication

Structure:

technology: (optional) *NX_CHAR*

Qualifier for the type of storage medium used.

Any of these values: `solid_state_disk` | `hard_disk` | `tape`

max_physical_capacity: (optional) *NX_NUMBER*

Total amount of data which the medium can hold.

name: (optional) *NX_CHAR*

Given name to the I/O unit.

FABRICATION: (optional) *NXfabrication*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcs_io_obj/FABRICATION-group*
- */NXcs_io_obj/max_physical_capacity-field*
- */NXcs_io_obj/name-field*
- */NXcs_io_obj/technology-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcs_io_obj.nxdl.xml

NXcs_io_sys

Status:

base class, extends *NXobject*

Description:

Computer science description of system of a computer.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXcs_io_obj

Structure:

CS_IO_OBJ: (optional) *NXcs_io_obj*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcs_io_sys/CS_IO_OBJ-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcs_io_sys.nxdl.xml

NXcs_mm_sys

Status:

base class, extends *NXobject*

Description:

Computer science description of a main memory system of a computer.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
none

Structure:

total_physical_memory: (optional) *NX_NUMBER*

How much physical memory does the system provide.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcs_mm_sys/total_physical_memory-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcs_mm_sys.nxdl.xml

NXcs_prng

Status:

base class, extends *NXobject*

Description:

Computer science description of pseudo-random number generator.

The purpose of such metadata is to identify if exactly the same sequence can be reproduced, like for a PRNG or not (for a true physically random source).

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
none

Structure:

type: (optional) *NX_CHAR*

Different approaches for generating random numbers with a computer exists. Some use a dedicated physical device where the state is unpredictable (physically). Some use a mangling of the system clock (system_clock), where also without additional pieces of information the sequence is not reproducible. Some use so-called pseudo-random number generator (PRNG) are used. These are algorithms which yield a deterministic sequence of practically randomly appearing numbers. These algorithms different in their quality in how close the resulting sequences are random. Nowadays one of the most commonly used algorithm is the MersenneTwister (mt19937).

Any of these values: `physical` | `system_clock` | `mt19937` | `other`

program: (optional) *NX_CHAR*

Name of the PRNG implementation and version. If such information is not available or if the PRNG type was set to other the DOI to the publication or the source code should be given.

@version: (optional) *NX_CHAR*

Version and build number, or commit hash.

seed: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

Parameter of the PRNG controlling its initialization and thus the specific sequence of numbers it generates.

warmup: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

Number of initial draws from the PRNG which are discarded in an effort to equilibrate the sequence and make it thus to statistically more random. If no warmup was performed or if warmup procedures are unclear, users should set the value to zero.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcs_prng/program-field*
- */NXcs_prng/program@version-attribute*
- */NXcs_prng/seed-field*
- */NXcs_prng/type-field*
- */NXcs_prng/warmup-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcs_prng.nxdl.xml

NXcs_profiling

Status:

base class, extends *NXobject*

Description:

Computer science description for summary performance/profiling data of an application.

Performance monitoring and benchmarking of software is a task where questions can be asked at various levels of detail. In general, there are three main contributions to performance:

- Hardware capabilities and configuration
- Software configuration and capabilities
- Dynamic effects of the system in operation and the system working together with eventually multiple computers, especially when these have to exchange information across a network.

At the most basic level users may wish to document how long e.g. a data analysis with a scientific software (app). A frequent idea is here to judge how critical the effect is on the workflow of the scientists, i.e. is the analysis possible in a few seconds or would it take days if I were to run this analysis on a comparable machine. In this case, mainly the order of magnitude is relevant, as well as how this can be achieved with using parallelization (i.e. reporting the number of CPU and GPU resources used, the number of processes and/or threads, and basic details about the computing node/computer.

At more advanced levels benchmarks may go as deep as detailed temporal tracking of individual processor instructions, their relation to other instructions, the state of call stacks, in short eventually the entire app execution history and hardware state history. Such analyses are mainly used for performance optimization as well as for tracking bugs and other development purposes. Specialized software exists which documents such performance data in specifically-formatted event log files or databases.

This base class cannot and should not replace these specific solutions. Instead, the intention of the base class is to serve scientists at the basic level to enable simple monitoring of performance data and log profiling data of key algorithmic steps or parts of computational workflows, so that these pieces of information can guide users which order of magnitude differences should be expected or not.

Developers of application definitions should add additional fields and references to e.g. more detailed performance data to which they wish to link the metadata in this base class.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXcs_computer, *NXcs_profiling_event*

Structure:

current_working_directory: (optional) *NX_CHAR*

Path to the directory from which the tool was called.

command_line_call: (optional) *NX_CHAR*

Command line call with arguments if applicable.

start_time: (optional) *NX_DATE_TIME*

ISO 8601 time code with local time zone offset to UTC information included when the app was started.

end_time: (optional) *NX_DATE_TIME*

ISO 8601 time code with local time zone offset to UTC information included when the app terminated or crashed.

total_elapsed_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

Wall-clock time how long the app execution took. This may be in principle end_time minus start_time; however usage of eventually more precise timers may warrant to use a finer temporal discretization, and thus demand a more precise record of the wall-clock time.

number_of_processes: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Qualifier which specifies with how many nominal processes the app was invoked. The main idea behind this field, for instance for app using a Message Passing Interface parallelization is to communicate how many processes were used.

For sequentially running apps number_of_processes and number_of_threads is 1. If the app uses exclusively GPU parallelization number_of_gpus can be larger than 1. If no GPU is used number_of_gpus is 0 even though the hardware may have GPUs installed, thus indicating these were not used though.

number_of_threads: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Qualifier with how many nominal threads were accessible to the app at runtime. Specifically here the maximum number of threads used for the high-level threading library used (e.g. OMP_NUM_THREADS), posix.

number_of_gpus: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Qualifier with how many nominal GPUs the app was invoked at runtime.

CS_COMPUTER: (optional) *NXcs_computer*

A collection with one or more computing nodes each with own resources. This can be as simple as a laptop or the nodes of a cluster computer.

CS_PROFILING_EVENT: (optional) *NXcs_profiling_event*

A collection of individual profiling event data which detail e.g. how much time the app took for certain computational steps and/or how much memory was consumed during these operations.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcs_profiling/command_line_call-field*
- */NXcs_profiling/CS_COMPUTER-group*
- */NXcs_profiling/CS_PROFILING_EVENT-group*
- */NXcs_profiling/current_working_directory-field*
- */NXcs_profiling/end_time-field*
- */NXcs_profiling/number_of_gpus-field*
- */NXcs_profiling/number_of_processes-field*
- */NXcs_profiling/number_of_threads-field*
- */NXcs_profiling/start_time-field*
- */NXcs_profiling/total_elapsed_time-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcs_profiling.nxdl.xml

NXcs_profiling_event

Status:

base class, extends *NXobject*

Description:

Computer science description of a profiling event.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_processes: Number of processes.

Groups cited:
none

Structure:

start_time: (optional) *NX_DATE_TIME*

ISO 8601 time code with local time zone offset to UTC information included when the event tracking started.

end_time: (optional) *NX_DATE_TIME*

ISO 8601 time code with local time zone offset to UTC information included when the event tracking ended.

description: (optional) *NX_CHAR*

Free-text description what was monitored/executed during the event.

elapsed_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

Wall-clock time how long the event took. This may be in principle end_time minus start_time; however usage of eventually more precise timers may warrant to use a finer temporal discretization, and thus demand a more precise record of the wall-clock time. Elapsed time may contain time portions where resources were idling.

number_of_processes: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Number of processes used (max) during the execution of this event.

number_of_threads: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Number of threads used (max) during the execution of this event.

number_of_gpus: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Number of GPUs used (max) during the execution of this event.

max_virtual_memory_snapshot: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_processes]) {units=*NX_ANY*}

Maximum amount of virtual memory allocated per process during the event.

max_resident_memory_snapshot: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_processes]) {units=*NX_ANY*}

Maximum amount of resident memory allocated per process during the event.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcs_profiling_event/description-field*
- */NXcs_profiling_event/elapsed_time-field*
- */NXcs_profiling_event/end_time-field*
- */NXcs_profiling_event/max_resident_memory_snapshot-field*
- */NXcs_profiling_event/max_virtual_memory_snapshot-field*
- */NXcs_profiling_event/number_of_gpus-field*
- */NXcs_profiling_event/number_of_processes-field*
- */NXcs_profiling_event/number_of_threads-field*
- */NXcs_profiling_event/start_time-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcs_profiling_event.nxdl.xml

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:

Any of these 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: `NXcxi_ptycho`

instrument_1: (required) *NXinstrument* <=

source_1: (required) *NXsource* <=

name: (required) *NX_CHAR* <=

energy: (required) *NX_FLOAT* <=

This is the energy of the machine, not the beamline.

probe: (required) *NX_FLOAT*

type: (required) *NX_FLOAT*

beam_1: (required) *NXbeam* <=

energy: (required) *NX_FLOAT*

@units: (required) *NX_CHAR*

extent: (optional) *NX_FLOAT* <=

@units: (required) *NX_CHAR*

incident_beam_divergence: (optional) *NX_FLOAT* <=

@units: (required) *NX_CHAR*

incident_beam_energy: (required) *NX_FLOAT*

@units: (required) *NX_CHAR*

incident_energy_spread: (required) *NX_FLOAT*

@units: (required) *NX_CHAR*

detector_1: (required) *NXdetector* <=

@axes: (required) *NX_CHAR*

should have value "[, data]"

@signal: (required) *NX_CHAR*

should have value "data"

translation: (required) *NX_FLOAT* {units=*NX_LENGTH*}

This is an array of shape (npts_x*npts_y, 3) and can be a Virtual Dataset of x and y

@units: (required) *NX_CHAR*

@axes: (required) *NX_CHAR*

this should take the value "translation:$slowaxisname:$fastaxisname"

@interpretation: (required) *NX_CHAR*

This should be "image"

data: (required) *NX_INT* (Rank: 3 for arbitrary scan, 4 for raster, Dimensions: [npts_x, npts_y, frame_size_x, frame_size_y])

x_pixel_size: (required) *NX_FLOAT* {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*} *<=*

@units: (required) *NX_CHAR*

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*
- */NXcxi_ptycho/entry_1/instrument_1/beam_1/incident_energy_spread-field*

- */NXcxi_ptycho/entry_1/instrument_1/beam_1/incident_energy_spread@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1-group*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/beam_center_x-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/beam_center_x@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/beam_center_y-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/beam_center_y@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/data-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/data_1-link*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/distance-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/distance@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/transformations-group*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/transformations/vector-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/translation-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/translation@axes-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/translation@interpretation-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/translation@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/x_pixel_size-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/x_pixel_size@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/y_pixel_size-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/y_pixel_size@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1@axes-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1@signal-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/MONITOR-group*
- */NXcxi_ptycho/entry_1/instrument_1/MONITOR/data-field*
- */NXcxi_ptycho/entry_1/instrument_1/source_1-group*
- */NXcxi_ptycho/entry_1/instrument_1/source_1/energy-field*
- */NXcxi_ptycho/entry_1/instrument_1/source_1/name-field*
- */NXcxi_ptycho/entry_1/instrument_1/source_1/probe-field*
- */NXcxi_ptycho/entry_1/instrument_1/source_1/type-field*
- */NXcxi_ptycho/entry_1/start_time-field*
- */NXcxi_ptycho/entry_1/title-field*
- */NXcxi_ptycho/sample_1-group*
- */NXcxi_ptycho/sample_1/geometry_1-group*
- */NXcxi_ptycho/sample_1/geometry_1/translation-link*
- */NXcxi_ptycho/sample_1/name-field*
- */NXcxi_ptycho/sample_1/transformations-group*

- */NXcxi_ptycho/sample_1/transformations@vector-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcxi_ptycho.nxdl.xml

NXdac

Status:

base class, extends *NXobject*

Description:

Digital-to-analog converter component/integrated circuit.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
none

Structure:

value: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

TBD.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXdac/value-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXdac.nxdl.xml

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

NXdelocalization

Status:

base class, extends *NXobject*

Description:

Base class to describe the delocalization of point-like objects on a grid.

Such a procedure is for instance used in image processing and e.g. atom probe microscopy (APM) to discretize a point cloud onto a grid to enable e.g. computing of point density, composition, or concentration values, obtain scalar fields, and compute gradients of these fields.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_p: Number of points/objects.

n_m: Number of mark data per point/object.

n_atoms: Number of atoms in the whitelist.

n_isotopes: Number of isotopes in the whitelist.

Groups cited:
none

Structure:

grid: (optional) *NX_CHAR*

Reference or link to the grid on which the delocalization is applied.

objects: (optional) *NX_CHAR*

Reference or link to the points which are delocalized on the grid.

weighting_model: (optional) *NX_CHAR*

The weighting model specifies how mark data are mapped to a weight per point. For atom probe microscopy (APM) as an example, different models are used which account differently for the multiplicity of an ion/atom:

- default, points all get the same weight 1.; for APM this is equivalent to ion species
- atomic_decomposition, points get as much weight as they have atoms of a type in element_whitelist,
- isotope_decomposition, points get as much weight as they have isotopes of a type in isotope_whitelist.

This description shows an example that could be reinterpreted for similar such data processing steps in other fields of science.

Any of these values:

- `default`
- `atomic_decomposition`
- `isotope_decomposition`

element_whitelist: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_atoms]) {units=*NX_UNITLESS*}

A list of elements (via proton number) to consider for the atomic_decomposition weighting model. Elements must exist in the periodic table of elements.

isotope_whitelist: (optional) *NX_UINT* (Rank: 2, Dimensions: [n_isotopes, 2]) {units=*NX_UNITLESS*}

A list of isotopes to consider for the isotope_decomposition weighting model. Isotopes must exist in the nuclid table. Entries in the fastest changing dimension should be the pair of proton (first entry) and neutron number (second entry).

mark: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_p, n_m])

Attribute data for each member of the point cloud. For APM these are the ion species labels generated via ranging. The number of mark data per point is 1 in the case for atom probe.

weight: (optional) *NX_NUMBER*

Weighting factor with which the integrated intensity per grid cell is multiplied specifically for each point. For APM the weight are positive integer values, specifically the multiplicity of the ion, according to the details of the weighting_model.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXdelocalization/element_whitelist-field*
- */NXdelocalization/grid-field*
- */NXdelocalization/isotope_whitelist-field*
- */NXdelocalization/mark-field*
- */NXdelocalization/objects-field*
- */NXdelocalization/weight-field*
- */NXdelocalization/weighting_model-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXdelocalization.nxdl.xml

NXdispersion

Status:

base class, extends *NXobject*

Description:

A dispersion denoting a sum of different dispersions. All NXdispersion_table and NXdispersion_function groups will be added together to form a single dispersion.

Symbols:

No symbol table

Groups cited:
NXdispersion_function, *NXdispersion_table*

Structure:

model_name: (optional) *NX_CHAR*

The name of the composite model.

DISPERSION_TABLE: (optional) *NXdispersion_table*

DISPERSION_FUNCTION: (optional) *NXdispersion_function*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXdispersion/DISPERSION_FUNCTION-group*
- */NXdispersion/DISPERSION_TABLE-group*
- */NXdispersion/model_name-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXdispersion.nxdl.xml

NXdispersion_function

Status:

base class, extends *NXobject*

Description:

This describes a dispersion function for a material or layer

Symbols:

n_repetitions: The number of repetitions for the repeated parameters

Groups cited:
NXdispersion_repeated_parameter, *NXdispersion_single_parameter*

Structure:

model_name: (optional) *NX_CHAR*

The name of this dispersion model.

formula: (optional) *NX_CHAR*

This should be a python parsable function. Here we should provide which keywords are available and a BNF of valid grammar.

convention: (optional) *NX_CHAR*

The sign convention being used (n + or - ik)

Any of these values: `n + ik` | `n - ik`

energy_identifier: (optional) *NX_CHAR*

The identifier used to represent energy in the formula. It is recommended to use *E*.

energy_min: (optional) *NX_NUMBER* {units=*NX_ENERGY*}

The minimum energy value at which this formula is valid.

energy_max: (optional) *NX_NUMBER* {units=*NX_ENERGY*}

The maximum energy value at which this formula is valid.

energy_unit: (optional) *NX_NUMBER* {units=*NX_ENERGY*}

The energy unit used in the formula. The field value is a scaling factor for the units attribute. It is recommeded to set the field value to 1 and carry all the unit scaling information in the units attribute.

wavelength_identifier: (optional) *NX_CHAR*

The identifier useed to represent wavelength in the formula. It is recommended to use *lambda*.

wavelength_unit: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The wavelength unit used in the formula. The field value is a scaling factor for the units attribute. It is recommeded to set the field value to 1 and carry all the unit scaling information in the units attribute.

wavelength_min: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The minimum wavelength value at which this formula is valid.

wavelength_max: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The maximum wavelength value at which this formula is valid.

representation: (optional) *NX_CHAR*

Which representation does the formula evaluate to. This may either be n for refractive index or eps for dielectric function. The appropriate token is then to be used inside the formula.

Any of these values: `n` | `eps`

DISPERSION_SINGLE_PARAMETER: (optional) *NXdispersion_single_parameter*

DISPERSION_REPEATED_PARAMETER: (optional) *NXdispersion_repeated_parameter*

parameter_units: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_repetitions]) <=

values: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_repetitions]) <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXdispersion_function/convention-field*
- */NXdispersion_function/DISPERSION_REPEATED_PARAMETER-group*
- */NXdispersion_function/DISPERSION_REPEATED_PARAMETER/parameter_units-field*
- */NXdispersion_function/DISPERSION_REPEATED_PARAMETER/values-field*
- */NXdispersion_function/DISPERSION_SINGLE_PARAMETER-group*
- */NXdispersion_function/energy_identifier-field*
- */NXdispersion_function/energy_max-field*
- */NXdispersion_function/energy_min-field*
- */NXdispersion_function/energy_unit-field*
- */NXdispersion_function/formula-field*
- */NXdispersion_function/model_name-field*
- */NXdispersion_function/representation-field*
- */NXdispersion_function/wavelength_identifier-field*

- */NXdispersion_function/wavelength_max-field*
- */NXdispersion_function/wavelength_min-field*
- */NXdispersion_function/wavelength_unit-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXdispersion_function.nxdl.xml

NXdispersion_repeated_parameter

Status:

base class, extends *NXobject*

Description:

A repeated parameter for a dispersion function

Symbols:

n_repetitions: The number of parameter repetitions

Groups cited:
none

Structure:

name: (optional) *NX_CHAR*

The name of the parameter

description: (optional) *NX_CHAR*

A description of what this parameter represents

parameter_units: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_repetitions])

A unit array associating a unit with each parameter. The first element should be equal to values/@unit. The values should be SI interpretable standard units with common prefixes (e.g. mikro, nano etc.) or their short-hand notation (e.g. nm, mm, kHz etc.).

values: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_repetitions]) {units=*NX_ANY*}

The value of the parameter

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXdispersion_repeated_parameter/description-field*
- */NXdispersion_repeated_parameter/name-field*
- */NXdispersion_repeated_parameter/parameter_units-field*
- */NXdispersion_repeated_parameter/values-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXdispersion_repeated_parameter.nxdl.xml

NXdispersion_single_parameter

Status:

base class, extends *NXobject*

Description:

A single parameter for a dispersion function

Symbols:

No symbol table

Groups cited:
none

Structure:

name: (optional) *NX_CHAR*

The name of the parameter

description: (optional) *NX_CHAR*

A description of what this parameter represents

value: (optional) *NX_NUMBER* {units=*NX_ANY*}

The value of the parameter

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXdispersion_single_parameter/description-field*
- */NXdispersion_single_parameter/name-field*
- */NXdispersion_single_parameter/value-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXdispersion_single_parameter.nxdl.xml

NXdispersion_table

Status:

base class, extends *NXobject*

Description:

A dispersion table denoting energy, dielectric function tabulated values.

Symbols:

The symbols in this schema to denote the dimensions

n_points: The number of energy and dielectric function points

Groups cited:
none

Structure:

model_name: (optional) *NX_CHAR*

The name of this dispersion model.

convention: (optional) *NX_CHAR*

The sign convention being used (n + or - ik)

Any of these values: `n + ik` | `n - ik`

wavelength: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_points]) {units=*NX_LENGTH*}

The wavelength array of the tabulated dataset. This is essentially a duplicate of the energy field. There should be one or both of them present.

energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_points]) {units=*NX_ENERGY*}

The energy array of the tabulated dataset. This is essentially a duplicate of the wavelength field. There should be one or both of them present.

refractive_index: (optional) *NX_COMPLEX* (Rank: 1, Dimensions: [n_points]) {units=*NX_DIMENSIONLESS*}

The refractive index array of the tabulated dataset.

dielectric_function: (optional) *NX_COMPLEX* (Rank: 1, Dimensions: [n_points]) {units=*NX_DIMENSIONLESS*}

The dielectric function of the tabulated dataset.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXdispersion_table/convention-field*
- */NXdispersion_table/dielectric_function-field*
- */NXdispersion_table/energy-field*
- */NXdispersion_table/model_name-field*
- */NXdispersion_table/refractive_index-field*
- */NXdispersion_table/wavelength-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXdispersion_table.nxdl.xml

NXdispersive_material

Status:

application definition, extends *NXobject*

Description:

NXdispersion

Symbols:

No symbol table

Groups cited:
NXcite, *NXdata*, *NXdispersion_function*, *NXdispersion_repeated_parameter*, *NXdispersion_single_parameter*, *NXdispersion_table*, *NXdispersion*, *NXentry*, *NXsample*

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR <=*

An application definition for a dispersive material.

Obligatory value: `NXdispersive_material` **@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

program: (recommended) *NX_CHAR*

Name of the program used for creating this dispersion

version: (recommended) *NX_CHAR*

Version of the program used

date: (recommended) *NX_DATE_TIME*

Date and time of creating this dispersion.

dispersion_type: (recommended) *NX_CHAR*

Denotes whether the dispersion is calculated or derived from an experiment

Any of these values: `measured` | `simulated`

sample: (required) *NXsample <=*

chemical_formula: (required) *NX_CHAR <=*

atom_types: (optional) *NX_CHAR*

List of comma-separated elements from the periodic table that are contained in the sample. If the sample substance has multiple components, all elements from each component must be included in *atom_types*.

colloquial_name: (optional) *NX_CHAR*

The colloquial name of the material, e.g. graphite or diamond for carbon.

material_phase: (optional) *NX_CHAR*

The phase of the material

Any of these values: `gas` | `liquid` | `solid` | `other`

material_phase_comment: (optional) *NX_CHAR*

Additional information about the phase if the material phase is other.

additional_phase_information: (recommended) *NX_CHAR*

This field may be used to denote additional phase information, such as crystalin phase of a crystal (e.g. 4H or 6H for SiC) or if a measurement was done on a thin film or bulk material.

REFERENCES: (recommended) *NXcite*

text: (required) *NX_CHAR*

A text description of this reference, e.g. *E. Example et al, The mighty example, An example journal 50 (2023), 100*

doi: (required) *NX_CHAR <=*

dispersion_x: (required) *NXdispersion*

The dispersion along the optical axis of the material. This should be the only dispersion available for isotropic materials. For uniaxial materials this denotes the ordinary axis. For biaxial materials this denotes the x axis or epsilon 11 tensor element of the diagionalized permittivity tensor. **model_name**: (required) *NX_CHAR <=*

The name of this dispersion model.

DISPERSION_TABLE: (recommended) *NXdispersion_table <=*

model_name: (required) *NX_CHAR <=*

convention: (required) *NX_CHAR <=*

wavelength: (required) *NX_NUMBER <=*

dielectric_function: (recommended) *NX_COMPLEX <=*

refractive_index: (recommended) *NX_COMPLEX <=*

DISPERSION_FUNCTION: (recommended) *NXdispersion_function <=*

model_name: (required) *NX_CHAR <=*

formula: (required) *NX_CHAR <=*

convention: (required) *NX_CHAR <=*

energy_identifier: (recommended) *NX_CHAR <=*

energy_unit: (recommended) *NX_NUMBER <=*

wavelength_identifier: (recommended) *NX_CHAR <=*

wavelength_unit: (recommended) *NX_NUMBER <=*

representation: (required) *NX_CHAR <=*

DISPERSION_SINGLE_PARAMETER: (required) *NXdispersion_single_parameter <=*

name: (required) *NX_CHAR <=*

value: (required) *NX_NUMBER <=*

DISPERSION_REPEATED_PARAMETER: (required) *NXdispersion_repeated_parameter <=*

name: (required) *NX_CHAR <=*

values: (required) *NX_NUMBER <=*

plot: (recommended) *NXdata*

dispersion_y: (optional) *NXdispersion*

This should only be filled for biaxial materials. It denotes the epsilon 22 direction of the diagionalized permittivity tensor. **model_name**: (required) *NX_CHAR <=*

The name of this dispersion model.

DISPERSION_TABLE: (recommended) *NXdispersion_table* <=

model_name: (required) *NX_CHAR* <=

convention: (required) *NX_CHAR* <=

wavelength: (required) *NX_NUMBER* <=

dielectric_function: (recommended) *NX_COMPLEX* <=

refractive_index: (recommended) *NX_COMPLEX* <=

DISPERSION_FUNCTION: (recommended) *NXdispersion_function* <=

model_name: (required) *NX_CHAR* <=

formula: (required) *NX_CHAR* <=

convention: (required) *NX_CHAR* <=

energy_identifier: (recommended) *NX_CHAR* <=

energy_unit: (recommended) *NX_NUMBER* <=

wavelength_identifier: (recommended) *NX_CHAR* <=

wavelength_unit: (recommended) *NX_NUMBER* <=

representation: (required) *NX_CHAR* <=

DISPERSION_SINGLE_PARAMETER: (required) *NXdispersion_single_parameter* <=

name: (required) *NX_CHAR* <=

value: (required) *NX_NUMBER* <=

DISPERSION_REPEATED_PARAMETER: (required) *NXdispersion_repeated_parameter* <=

name: (required) *NX_CHAR* <=

values: (required) *NX_NUMBER* <=

plot: (recommended) *NXdata*

dispersion_z: (optional) *NXdispersion*

This should only be filled for uniaxial or biaxial materials. For uniaxial materials this denotes the extraordinary axis. For biaxial materials this denotes the epsilon 33 tensor element of the diagionalized perimittivty tensor. **model_name**: (required) *NX_CHAR* <=

The name of this dispersion model.

DISPERSION_TABLE: (recommended) *NXdispersion_table* <=

model_name: (required) *NX_CHAR* <=

convention: (required) *NX_CHAR* <=

wavelength: (required) *NX_NUMBER* <=

dielectric_function: (recommended) *NX_COMPLEX* <=

refractive_index: (recommended) *NX_COMPLEX* <=

DISPERSION_FUNCTION: (recommended) *NXdispersion_function* <=

model_name: (required) *NX_CHAR* <=

formula: (required) *NX_CHAR* <=

convention: (required) *NX_CHAR* <=

energy_identifier: (recommended) *NX_CHAR* <=

energy_unit: (recommended) *NX_NUMBER* <=

wavelength_identifier: (recommended) *NX_CHAR* <=

wavelength_unit: (recommended) *NX_NUMBER* <=

representation: (required) *NX_CHAR* <=

DISPERSION_SINGLE_PARAMETER: (required) *NXdispersion_single_parameter* <=

name: (required) *NX_CHAR* <=

value: (required) *NX_NUMBER* <=

DISPERSION_REPEATED_PARAMETER: (required) *NXdispersion_repeated_parameter* <=

name: (required) *NX_CHAR* <=

values: (required) *NX_NUMBER* <=

plot: (recommended) *NXdata*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXdispersive_material/ENTRY-group*
- */NXdispersive_material/ENTRY/date-field*
- */NXdispersive_material/ENTRY/definition-field*
- */NXdispersive_material/ENTRY/definition@url-attribute*
- */NXdispersive_material/ENTRY/definition@version-attribute*
- */NXdispersive_material/ENTRY/dispersion_type-field*
- */NXdispersive_material/ENTRY/dispersion_x-group*
- */NXdispersive_material/ENTRY/dispersion_x/DISPERSION_FUNCTION-group*
- */NXdispersive_material/ENTRY/dispersion_x/DISPERSION_FUNCTION/convention-field*
- */NXdispersive_material/ENTRY/dispersion_x/DISPERSION_FUNCTION/DISPERSION_REPEATED_PARAMETER-group*
- */NXdispersive_material/ENTRY/dispersion_x/DISPERSION_FUNCTION/DISPERSION_REPEATED_PARAMETER/name-field*
- */NXdispersive_material/ENTRY/dispersion_x/DISPERSION_FUNCTION/DISPERSION_REPEATED_PARAMETER/values-field*
- */NXdispersive_material/ENTRY/dispersion_x/DISPERSION_FUNCTION/DISPERSION_SINGLE_PARAMETER-group*

- */NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/wavelength_identifier-field*
- */NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/wavelength_unit-field*
- */NXdispersive_material/ENTRY/dispersion_y/DISPERSION_TABLE-group*
- */NXdispersive_material/ENTRY/dispersion_y/DISPERSION_TABLE/convention-field*
- */NXdispersive_material/ENTRY/dispersion_y/DISPERSION_TABLE/dielectric_function-field*
- */NXdispersive_material/ENTRY/dispersion_y/DISPERSION_TABLE/model_name-field*
- */NXdispersive_material/ENTRY/dispersion_y/DISPERSION_TABLE/refractive_index-field*
- */NXdispersive_material/ENTRY/dispersion_y/DISPERSION_TABLE/wavelength-field*
- */NXdispersive_material/ENTRY/dispersion_y/model_name-field*
- */NXdispersive_material/ENTRY/dispersion_y/plot-group*
- */NXdispersive_material/ENTRY/dispersion_z-group*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION-group*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/convention-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/DISPERSION_REPEATED_PARAMETER-group*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/DISPERSION_REPEATED_PARAMETER/name-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/DISPERSION_REPEATED_PARAMETER/values-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/DISPERSION_SINGLE_PARAMETER-group*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/DISPERSION_SINGLE_PARAMETER/name-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/DISPERSION_SINGLE_PARAMETER/value-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/energy_identifier-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/energy_unit-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/formula-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/model_name-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/representation-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/wavelength_identifier-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/wavelength_unit-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_TABLE-group*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_TABLE/convention-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_TABLE/dielectric_function-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_TABLE/model_name-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_TABLE/refractive_index-field*
- */NXdispersive_material/ENTRY/dispersion_z/DISPERSION_TABLE/wavelength-field*

- */NXdispersive_material/ENTRY/dispersion_z/model_name-field*
- */NXdispersive_material/ENTRY/dispersion_z/plot-group*
- */NXdispersive_material/ENTRY/program-field*
- */NXdispersive_material/ENTRY/REFERENCES-group*
- */NXdispersive_material/ENTRY/REFERENCES/doi-field*
- */NXdispersive_material/ENTRY/REFERENCES/text-field*
- */NXdispersive_material/ENTRY/sample-group*
- */NXdispersive_material/ENTRY/sample/additional_phase_information-field*
- */NXdispersive_material/ENTRY/sample/atom_types-field*
- */NXdispersive_material/ENTRY/sample/chemical_formula-field*
- */NXdispersive_material/ENTRY/sample/colloquial_name-field*
- */NXdispersive_material/ENTRY/sample/material_phase-field*
- */NXdispersive_material/ENTRY/sample/material_phase_comment-field*
- */NXdispersive_material/ENTRY/version-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXdispersive_material.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 beam in electron microscopy.

Symbols:

No symbol table

Groups cited:

NXaperture_em, *NXbeam*, *NXcorrector_cs*, *NXfabrication*, *NXlens_em*, *NXsensor*, *NXsource*, *NXstage_lab*, *NXtransformations*

Structure:

FABRICATION: (optional) *NXfabrication*

electron_source: (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.

FABRICATION: (optional) *NXfabrication*

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 ocharacterization 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_source-group*
- */NXebeam_column/electron_source/description-field*
- */NXebeam_column/electron_source/emitter_material-field*
- */NXebeam_column/electron_source/emitter_type-field*
- */NXebeam_column/electron_source/FABRICATION-group*
- */NXebeam_column/electron_source/name-field*
- */NXebeam_column/electron_source/probe-field*
- */NXebeam_column/electron_source/TRANSFORMATIONS-group*
- */NXebeam_column/electron_source/voltage-field*
- */NXebeam_column/FABRICATION-group*
- */NXebeam_column/LENS_EM-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 symultaneously, without scanning a pysical quantity)

nsa: Number of slow axes (axes acquired scanning a pysical 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.

Table 1: Examples

Mode	fast_axes	slow_axes
Hemispherical in ARPES mode	[‘energy’, ‘kx’]	
Hemispherical with channeltron, sweeping energy mode		["energy"]
Tof	[‘energy’, ‘kx’, ‘ky’]	
Momentum microscope, spin-resolved	[‘energy’, ‘kx’, ‘ky’]	[‘spin up-down’, ‘spin left-right’]

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 positon 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 electron analyser 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.

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/SPINDISPERSION-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:

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*}

volage 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 *NXopt*

Description:

Ellipsometry, complex systems, up to variable angle spectroscopy.

Information on ellipsometry is provided, e.g. in:

- H. Fujiwara, Spectroscopic ellipsometry: principles and applications, John Wiley & Sons, 2007.
- R. M. A. Azzam and N. M. Bashara, Ellipsometry and Polarized Light, North-Holland Publishing Company, 1977.
- H. G. Tompkins and E. A. Irene, Handbook of Ellipsometry, William Andrew, 2005.

Open access sources:

- https://www.angstromadvanced.com/resource.asp
- https://pypolar.readthedocs.io/en/latest/

Review articles:

- T. E. Jenkins, "Multiple-angle-of-incidence ellipsometry", J. Phys. D: Appl. Phys. 32, R45 (1999), https://doi.org/10.1088/0022-3727/32/9/201
- D. E. Aspnes, "Spectroscopic ellipsometry - Past, present, and future", Thin Solid Films 571, 334-344 (2014), https://doi.org/10.1016/j.tsf.2014.03.056
- R. M. A. Azzam, "Mueller-matrix ellipsometry: a review", Proc. SPIE 3121, Polarization: Measurement, Analysis, and Remote Sensing, (3 October 1997), https://doi.org/10.1117/12.283870
- E. A. Irene, "Applications of spectroscopic ellipsometry to microelectronics", Thin Solid Films 233, 96-111 (1993), https://doi.org/10.1016/0040-6090(93)90069-2
- S. Zollner et al., "Spectroscopic ellipsometry from 10 to 700 K", Adv. Opt. Techn., (2022), https://doi.org/10.1515/aot-2022-0016

Symbols:

Variables used throughout the document, e.g. dimensions or parameters.

N_spectrum: Length of the spectrum array (e.g. wavelength or energy) of the measured data.

N_sensors: Number of sensors used to measure parameters that influence the sample, such as temperature or pressure.

N_measurements: Number of measurements (1st dimension of measured_data array). This is equal to the number of parameters scanned. For example, if the experiment was performed at three different temperatures and two different pressures N_measurements = 2*3 = 6.

N_detection_angles: Number of detection angles of the beam reflected or scattered off the sample.

N_incident_angles: Number of angles of incidence of the incident beam.

N_observables: Number of observables that are saved in a measurement. e.g. one for intensity, reflectivity or transmittance, two for Psi and Delta etc. This is equal to the second dimension of the data array 'measured_data' and the number of column names.

N_time: Number of time points measured, the length of NXsample/time_points

Groups cited:
NXbeam_path, *NXdetector*, *NXentry*, *NXinstrument*, *NXlens_opt*, *NXmonochromator*, *NXprocess*, *NXsample*, *NXsource*, *NXwaveplate*

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
- 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_description: (required) *NX_CHAR <=*

An optional free-text description of the experiment.

However, details of the experiment should be defined in the specific fields of this application definition rather than in this experiment description.

experiment_type: (required) *NX_CHAR <=*

Specify the type of ellipsometry.

Any of these values:

- `in situ spectroscopic ellipsometry`
- `THz spectroscopic ellipsometry`
- `infrared spectroscopic ellipsometry`
- `ultraviolet spectroscopic ellipsometry`
- `uv-vis spectroscopic ellipsometry`
- `NIR-Vis-UV spectroscopic ellipsometry`
- `imaging ellipsometry`

INSTRUMENT: (required) *NXinstrument <=*

Properties of the ellipsometry equipment.

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.

ellipsometer_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`

rotating_element_type: (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)`

software: (required) *NXprocess <=*

program: (required) *NX_CHAR <=*

Commercial or otherwise defined given name of 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.

BEAM_PATH: (required) *NXbeam_path <=*

light_source: (required) *NXsource <=*

Specify the used light source. Multiple selection possible.

source_type: (required) *NX_CHAR*

Any of these values: `arc lamp` | `halogen lamp` | `LED` | `other`

focussing_probes: (optional) *NXlens_opt*

If focussing probes (lenses) were used, please state if the data were corrected for the window effects. **data_correction**: (required) *NX_BOOLEAN*

Were the recorded data corrected by the window effects of the focussing probes (lenses)?

angular_spread: (recommended) *NX_NUMBER* {units=*NX_ANGLE*}

Specify the angular spread caused by the focussing probes.

DETECTOR: (required) *NXdetector*

Properties of the detector used. Integration time is the count time field, or the real time field. See their definition.

rotating_element: (optional) *NXwaveplate <=*

Properties of the rotating element defined in 'instrument/rotating_element_type'. **revolutions**: (optional) *NX_NUMBER* {units=*NX_COUNT*}

Define how many revolutions of the rotating element were averaged for each measurement. If the number of revolutions was fixed to a certain value use the field 'fixed_revolutions' instead.

fixed_revolutions: (optional) *NX_NUMBER* {units=*NX_COUNT*}

Define how many revolutions of the rotating element were taken into account for each measurement (if number of revolutions was fixed to a certain value, i.e. not averaged).

max_revolutions: (optional) *NX_NUMBER* {units=*NX_COUNT*}

Specify the maximum value of revolutions of the rotating element for each measurement.

spectrometer: (optional) *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.

SAMPLE: (required) *NXsample <=*

backside_roughness: (required) *NX_BOOLEAN*

Was the backside of the sample roughened? Relevant for infrared ellipsometry.

data_collection: (required) *NXprocess <=*

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_observables) 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`

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXellipsometry/ENTRY-group*
- */NXellipsometry/ENTRY/data_collection-group*
- */NXellipsometry/ENTRY/data_collection/data_type-field*
- */NXellipsometry/ENTRY/definition-field*
- */NXellipsometry/ENTRY/definition@url-attribute*
- */NXellipsometry/ENTRY/definition@version-attribute*
- */NXellipsometry/ENTRY/experiment_description-field*
- */NXellipsometry/ENTRY/experiment_type-field*
- */NXellipsometry/ENTRY/INSTRUMENT-group*
- */NXellipsometry/ENTRY/INSTRUMENT/BEAM_PATH-group*
- */NXellipsometry/ENTRY/INSTRUMENT/BEAM_PATH/DETECTOR-group*
- */NXellipsometry/ENTRY/INSTRUMENT/BEAM_PATH/focussing_probes-group*
- */NXellipsometry/ENTRY/INSTRUMENT/BEAM_PATH/focussing_probes/angular_spread-field*
- */NXellipsometry/ENTRY/INSTRUMENT/BEAM_PATH/focussing_probes/data_correction-field*
- */NXellipsometry/ENTRY/INSTRUMENT/BEAM_PATH/light_source-group*
- */NXellipsometry/ENTRY/INSTRUMENT/BEAM_PATH/light_source/source_type-field*
- */NXellipsometry/ENTRY/INSTRUMENT/BEAM_PATH/rotating_element-group*
- */NXellipsometry/ENTRY/INSTRUMENT/BEAM_PATH/rotating_element/fixed_revolutions-field*
- */NXellipsometry/ENTRY/INSTRUMENT/BEAM_PATH/rotating_element/max_revolutions-field*
- */NXellipsometry/ENTRY/INSTRUMENT/BEAM_PATH/rotating_element/revolutions-field*
- */NXellipsometry/ENTRY/INSTRUMENT/BEAM_PATH/spectrometer-group*
- */NXellipsometry/ENTRY/INSTRUMENT/company-field*
- */NXellipsometry/ENTRY/INSTRUMENT/construction_year-field*
- */NXellipsometry/ENTRY/INSTRUMENT/ellipsometer_type-field*
- */NXellipsometry/ENTRY/INSTRUMENT/rotating_element_type-field*
- */NXellipsometry/ENTRY/INSTRUMENT/software-group*
- */NXellipsometry/ENTRY/INSTRUMENT/software/program-field*
- */NXellipsometry/ENTRY/SAMPLE-group*
- */NXellipsometry/ENTRY/SAMPLE/backside_roughness-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXellipsometry.nxdl.xml

NXem

Status:

application definition, extends *NXobject*

Description:

Characterization of a sample during a session on an electron microscope.

The idea and aim of NXem: Electron microscopy (EM) research, whether it be performed with scanning electron microscope (SEM) or transmission electron microscope (TEM) instruments, uses versatile tools for preparing and characterizing samples and specimens. The term specimen is considered a synonym for sample in this application definition. A specimen is a physical portion of material that is studied/characterized during the microscope session, eventually in different places on the specimen surface, illuminating the surface layers or shining through thin specimens. These places are named regions of interest (ROIs).

Fundamentally, an electron microscope is an electron accelerator. Experimentalists use it 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. The application definition focuses on the usage of EM in materials research. The application definition design makes it in principle applicable also in cryo-EM on biomaterials.

Multiple specimens have to be described with multiple *NXentry* instances.

Electron microscopes motivate the development of a comprehensive data schema: 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 scientists. These users may perform analyses for other users as a service task, especially in large research facility settings. Oftentimes, though, and especially for cutting-edge instruments, the scientists guide the process and maybe even control the microscope. Instruments are usually controlled on-premises but also more and more functionalities for remote control have become available. Scientists oftentimes 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 making separate schemas for SEM or TEM are primarily the key similarities of SEM and TEM instruments:

Both type of instruments have electro-magnetic lenses. These may differ in design, alignment, number, and level of corrected for aberrations. As an obvious difference, a TEM is mainly used for measuring the transmitted electron beam. This calls for using a different lens setup and relative placement of the specimen in the lens setup. Also TEM specimens are substantially thinner than specimens characterized with SEM to enable an illumination through the specimen. This offers capabilities for probing of additional physical mechanisms of electron-matter interaction which are unavailable in SEMs.

Nevertheless, both types of electron microscopes use detector systems which measure different types of signals that originate from the same set of radiation/specimen interactions. Often these detectors have a similar design and technology or are even used both in SEMs and TEMs.

A comprehensive schema instead of specific SEM or TEM schemas: 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 it is the addressed circle of users which develops or uses schemas for research data management (RDM) with EM, the more understandable it is that scientists of either community (or sub-community) ask for designing method-specific schemas.

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 schemas such as NXem_ebsd, NXem_eels, NXem_edx, or NXem_imaging.

However, the history of electron microscopy has shown that these activities led to a zoo of schemas and vocabulary, with implementation in many data and file formats, difficult to make interoperable. Instead of trying to maintain this, we would like to advocate that the FAIR principles should 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 schemas with associated representations of data and metadata: Each combination of schemas 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 have to be maintained. 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.

The advantage of working towards a common vocabulary and representation: A common vocabulary can serve interoperability as developers of schemas and scientists can reuse for instance these terms, thus supporting interoperability. Ideally, scientists specialize an 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 schemas.

Aiming for more standardization, i.e. a lower number of schemas rather than a single standard for electron microscopy is a compromise that can serve academia and industry as it enables a focusing of software development efforts on those schemas, on fixing and discussing them, and on harmonizing their common vocabulary. These activities can be specifically relevant also for technology partners building EM hard- and software as it improves the longevity of certain schemas; and thus can help with incentivizing them to support the community with implementing support for such schemas into their 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 defining schemas for EM research. Thereby, NXem can be useful to extends top-level ontologies towards a domain- and method-specific ontology for electron microscopy as it is used for materials research.

Working towards a common vocabulary is a community activity that profits from everybody reflecting in detail whether certain terms they have used in the past 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.

Addressing the generality versus specificity challenge: It is noteworthy to understand that (not only for NeXus), schemas differ already if at least one field is required in one version of the schema, but it is set optional in another schema. If group(s), field(s), or attributes are removed or added, or even a docstring is changed, schemas can become inconsistent. It is noteworthy to mention that the idea of a NeXus application definition serves as a contract between a data provider and a data consumer. Providers can be the software of a specific microscopes or users with specific analysis needs. Consumers can be again specific software tools, like vendor software for controlling the instrument or a scientific software for doing artificial intelligence analyses on EM data). Such changes of a schema lead to new versions.

Verification of constraints and conditions: Tools like NeXus do not avoid or protect against all such possible inconsistencies; however NeXus offers a mechanism and toolset, through which schemas 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.

This stresses again the fundamental and necessary role of working towards a common vocabulary and, with a longer perspective in mind, a machine-actionable knowledge representation and verification engine. So far many conditions and requirements are formulated in the docstrings of the respective entries of the application definition.

NXem takes a key step towards standardization of EM data schemas. It offers a controlled vocabulary and set of relations between concepts and enables the description of the data which are collected for research with electron microscopes. To be most efficient and offering reusability, the NXem application definition should be understood as a template that one should ideally use as is. NXem can be considered a base for designing more specialized definitions. These should ideally be prefixed with NXem_method (e.g. NXem_ebsd).

The use of NXem should be as follows: Offspring application definitions should not remove groups but leave these optional or, even better, propose changes to NXem.

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. It is questionable to which level such a reproducibility is possible with EM at all considering beam damage, effects of the environment, and other not exactly quantifiable influences. While this points to the physical limitations there are also practical and economical constraints on how completely EM research can be documented: 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 as technology partners have a substantiated interest in finding a compromise between being open to their users and securing their business models.

By design, EM experiments 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 but being system-agnostic at the same time. Settings of apertures are an example where aperture modes are in most cases aliases behind which there is a set of very detailed settings specific to the software and control units used. These settings are difficult to retrieve, are not fully documented by technology partners. This simplification for users of microscopes makes experiments easier understandable. On the flipside these subtilities limit the opportunities of especially open- source developments to make data schemas covering enough for general usage and specific enough and sufficiently detailed to remain useful for research by electron microscopy domain experts.

Instead, currently it is for the docstring to specify what is conceptually eventually behind such aliases. The design rule we followed while drafting this NXem 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 as discussed 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 is when the application definition is expected to hold all data that might be of relevance for future questions.

We are skeptical if there is one such representation that can fulfill all these aims and interest, while remaining at the same time approachable and executable by a large number of scientists in a community. However, we are also convinced that this is not a reason to accept the status quo of having a very large set of oftentimes strongly overlapping and redundant schemas.

NXem is our proposal to motivate the EM community to work towards more standardization and discussion of what constitutes data, i.e. metadata, numerical and categorical data in research with electron microscopes. 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. Therefore,

This application definition has the following components at the top-level:

- Each signal, such as a spectrum or image taken at the microscope, should have an associated time-zone-aware time stamp and report of the specific settings of the microscope at that point in time when the image was taken. This is why instances of *NXevent_data_em* have their own em_lab section. The reason is that EMs can be highly dynamic, 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 scans and/or surface mappings in an SEM, or the ion-beam-milling of a specimen in preparation for e.g. 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 time-zone-aware dates of the components, which is relevant for documenting as exactly as practically possible settings when images and spectra were taken.
- *NXinstrument*; conceptually this is a container to store an arbitrary level of detail of the technical components of the microscope as a device and the lab in which the microscope 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 of who executed an event of data acquisition or processing.
- *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*; at the top-level, 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. Default plottables are not intended to serve every possible analysis and visualization demand but are instead a preview. We made this choice because what constitutes a useful default plot is often 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. 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:

- Images and spectra should be stored as *NXimage_set* and *NXspectrum_set* instances to hold data at the earliest possible step in the computational processing (which is usually performed with the microscope control and or integrated analysis software). The formatting of the NXdata groups enables the display of image and spectra with web technology visualization software.
- When two- and three-dimensional geometric primitive data are stored, it is useful to write additional optional XDMF fields which support additional plotting of the data with visualization software.
- 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 based on electron counts are typically detected with specific operation modes such as bright field or dark field imaging in TEM or secondary/back-scattered electron imaging in SEM.
- Also spectra (X-ray quanta or Auger electron counts) typically are referred to under the assumption of a specific operation mode of the microscope.
- These data are in virtually all cases a result of some numerical processing. These data and processing steps are modelled as instances of *NXprocess* which use terms from a controlled vocabulary e.g. SE (secondary electron), BSE (back-scattered electron), Kikuchi, X-ray, Auger, Cathodolum(inescence).

A key question often asked with EM experiments is how the actual (meta)data should be stored (in memory or on disk).

The application definition NXem is a graph which describes how numerical data and (meta)data for EM research are related to one another.

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.

Symbols:

No symbol table

Groups cited:

NXaberration_model_ceos, *NXaberration_model_nion*, *NXaberration*, *NXaperture_em*, *NXchamber*, *NXcoordinate_system_set*, *NXcorrector_cs*, *NXdata*, *NXdetector*, *NXebeam_column*, *NXentry*, *NXevent_data_em_set*, *NXevent_data_em*, *NXfabrication*, *NXibeam_column*, *NXimage_set*, *NXinstrument*, *NXion*, *NXlens_em*, *NXmonitor*, *NXnote*, *NXoptical_system_em*, *NXprocess*, *NXprogram*, *NXpump*, *NXsample*, *NXscanbox_em*, *NXsource*, *NXspectrum_set*, *NXstage_lab*, *NXtransformations*, *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: (recommended) *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 by specifying both a start_time and an 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_data_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) *NXprogram*

program: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

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 <=*

USER: (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: (recommended) *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. If this field is field the specific service should also be written in orcid_platform

orcid_platform: (recommended) *NX_CHAR* <=

Name of the OrcID or ResearcherID where the account under orcid is registered.

telephone_number: (optional) *NX_CHAR* <=

(Business) (tele)phone number of the user at the point in time when the experiment was performed.

role: (recommended) *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: `experiment` | `simulation`

name: (required) *NX_CHAR* <=

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 for this, more convenient alias name.

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 analysis.

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*

List of comma-separated elements from the periodic table that are contained in the sample. If the sample substance has multiple components, all elements from each component must be included in *atom_types*.

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. For scanning electron microscopy, in many cases the specimen is much thicker than what is illuminatable by the electron beam. In this case the value should be set to the actual thickness of the specimen viewed for an illumination situation where the nominal surface normal of the specimen is parallel to the optical axis.

density: (optional) *NX_FLOAT* {units=*NX_ANY*} <=

(Measured) density of the specimen. For multi-layered specimens this field should only be used to describe the density of the excited volume. For scanning electron microscopy the usage of this field is discouraged and instead an instance of an *NXinteraction_vol_em* within individual *NXevent_data_em* instances can provide a much better description of the relevant details why one may wish to store the density of the specimen.

description: (optional) *NX_CHAR <=*

Discouraged free-text field in case properly designed records for the sample_history are not available.

DATA: (optional) *NXdata <=*

Hard link to a location in the hierarchy of the NeXus file where the data for default plotting are stored.

COORDINATE_SYSTEM_SET: (recommended) *NXcoordinate_system_set*

TRANSFORMATIONS: (optional) *NXtransformations <=*

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 NXevent_data_em sections. These event instances take the role of time snapshot. For an NXevent_data_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_source 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_source 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.

FABRICATION: (required) *NXfabrication*

vendor: (recommended) *NX_CHAR <=*

model: (recommended) *NX_CHAR <=*

identifier: (recommended) *NX_CHAR* <=

capabilities: (optional) *NX_CHAR*

CHAMBER: (optional) *NXchamber*

EBEAM_COLUMN: (required) *NXebeam_column*

FABRICATION: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

CHAMBER: (optional) *NXchamber*

electron_source: (required) *NXsource* <=

name: (recommended) *NX_CHAR* <=

voltage: (required) *NX_FLOAT* <=

emitter_type: (recommended) *NX_CHAR* <=

Any of these values: `thermionic` | `schottky` | `field_emission`

FABRICATION: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

APERTURE_EM: (optional) *NXaperture_em* <=

value: (required) *NX_NUMBER* <=

name: (required) *NX_CHAR* <=

description: (optional) *NX_CHAR* <=

FABRICATION: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

LENS_EM: (optional) *NXlens_em* <=

If the lens is described at least one of the fields voltage, current, or value should be defined. **voltage**: (recommended) *NX_NUMBER* <=

current: (recommended) *NX_NUMBER* <=

value: (recommended) *NX_NUMBER* <=

FABRICATION: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

aberration_correction: (recommended) *NXcorrector_cs*

applied: (required) *NX_BOOLEAN* <=

name: (optional) *NX_CHAR* <=

FABRICATION: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

ZEMLIN_TABLEAU: (recommended) *NXprocess* <=

description: (optional) *NX_CHAR* <=

tilt_angle: (recommended) *NX_FLOAT* {units=*NX_ANGLE*} <=

exposure_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

magnification: (optional) *NX_NUMBER* {units=*NX_DIMENSIONLESS*} <=

PROCESS: (optional) *NXprocess* <=

ceos: (optional) *NXaberration_model_ceos* <=

c_1: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

a_1: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

b_2: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

a_2: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

c_3: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

s_3: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

a_3: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

b_4: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

d_4: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

a_4: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

c_5: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

s_5: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

r_5: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

a_6: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

nion: (optional) *NXaberration_model_nion* <=

c_1_0: (recommended) *NXaberration* <=

- **magnitude**: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=
- **uncertainty**: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=
- **uncertainty_model**: (recommended) *NX_CHAR* <=
- **delta_time**: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_1_2_a: (recommended) *NXaberration* <=

- **magnitude**: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=
- **uncertainty**: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=
- **uncertainty_model**: (recommended) *NX_CHAR* <=
- **delta_time**: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_1_2_b: (recommended) *NXaberration* <=

- **magnitude**: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=
- **uncertainty**: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=
- **uncertainty_model**: (recommended) *NX_CHAR* <=
- **delta_time**: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_2_1_a: (recommended) *NXaberration* <=

- **magnitude**: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=
- **uncertainty**: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=
- **uncertainty_model**: (recommended) *NX_CHAR* <=
- **delta_time**: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_2_1_b: (recommended) *NXaberration* <=

- **magnitude**: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=
- **uncertainty**: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=
- **uncertainty_model**: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_2_3_a: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_2_3_b: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_3_0: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_3_2_a: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_3_2_b: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_3_4_a: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_3_4_b: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_4_1_a: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_4_1_b: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_4_3_a: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_4_3_b: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_4_5_a: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_4_5_b: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_5_0: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_5_2_a: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_5_2_b: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_5_4_a: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_5_4_b: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_5_6_a: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_5_6_b: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

IBEAM_COLUMN: (optional) *NXibeam_column*

FABRICATION: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

CHAMBER: (optional) *NXchamber*

ion_source: (required) *NXsource* <=

name: (recommended) *NX_CHAR* <=

emitter_type: (required) *NX_CHAR* <=

voltage: (required) *NX_FLOAT* <=

current: (required) *NX_FLOAT* <=

FABRICATION: (recommended) *NXfabrication*

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

probe: (required) *NXion* <=

charge_state: (recommended) *NX_INT* <=

isotope_vector: (recommended) *NX_UINT* <=

name: (recommended) *NX_CHAR* <=

APERTURE_EM: (recommended) *NXaperture_em* <=

value: (required) *NX_NUMBER* <=

FABRICATION: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

LENS_EM: (recommended) *NXlens_em* <=

If the lens is described at least one of the fields voltage, current, or value should be defined. **voltage**: (recommended) *NX_NUMBER* <=

current: (recommended) *NX_NUMBER* <=

value: (recommended) *NX_NUMBER* <=

FABRICATION: (recommended) *NXfabrication* <=

identifier: (recommended) *NX_CHAR* <=

EBEAM_DEFLECTOR: (optional) *NXscanbox_em*

pixel_time: (recommended) *NX_FLOAT* <=

FABRICATION: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

IBEAM_DEFLECTOR: (optional) *NXscanbox_em*

FABRICATION: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

OPTICAL_SYSTEM_EM: (recommended) *NXoptical_system_em*

camera_length: (optional) *NX_NUMBER* <=

magnification: (optional) *NX_NUMBER* <=

defocus: (recommended) *NX_NUMBER* <=

semi_convergence_angle: (recommended) *NX_NUMBER* <=

working_distance: (recommended) *NX_FLOAT* <=

beam_current: (recommended) *NX_FLOAT* <=

beam_current_description: (recommended) *NX_CHAR* <=

DETECTOR: (required) *NXdetector* <=

Description of the type of the detector.

Electron microscopes have typically multiple detectors. Different technologies are in use like CCD, scintillator, direct electron, CMOS, or image plate to name but a few. **local_name**: (required) *NX_CHAR* <=

Instrument-specific alias/name

FABRICATION: (recommended) *NXfabrication*

identifier: (recommended) *NX_CHAR* <=

PUMP: (optional) *NXpump*

stage_lab: (required) *NXstage_lab*

name: (required) *NX_CHAR <=*

design: (recommended) *NX_CHAR <=*

description: (optional) *NX_CHAR <=*

position: (recommended) *NX_FLOAT <=*

rotation: (recommended) *NX_FLOAT <=*

tilt_1: (recommended) *NX_FLOAT <=*

tilt_2: (recommended) *NX_FLOAT <=*

FABRICATION: (recommended) *NXfabrication <=*

vendor: (recommended) *NX_CHAR <=*

model: (recommended) *NX_CHAR <=*

identifier: (recommended) *NX_CHAR <=*

capabilities: (optional) *NX_CHAR*

measurement: (optional) *NXevent_data_em_set*

A container for storing a set of NXevent_data_em instances.

An event is a time interval during which the microscope was configured in a specific way. The microscope is considered as stable enough during this interval that a measurement with one or multiple detectors is possible. What constitutes such time interval depends on how the microscope is used and which measurements the user would like to perform.

Each NXevent_data_em instance holds one acquisition task with detectors. Each NXevent_data_em section contains an em_lab group in which specific settings and states of the microscope during the time interval can be stored to document the history of states of the microscope over the course of the session.

The NXem application definition offers maximal flexibility. One extreme could be that the only one NXevent_data_em instance is used and this covers the time interval of the entire session at the microscope. The other extreme case is that each collection of an image or even single spot measurement is defined as an NXevent_data_em instance. In this case the em_lab group inside the NXevent_data_em also holds the specific time-dependent state of the microscope with which in theory all dynamics of the system (if measured) can be captured and documented.

Nowadays microscopes exists for which hard- and software solutions enable a tracking of the dynamics of the microscope and the actions of the user (such as with solution like AXONSynchronicity from Protochips). The NXem application definition can however also be used for less complex interaction and lower demands wrt to time tracking activities.

An NXevent_data_em instance holds specific details about how raw data from a detector were processed. Raw data may already be post-processed as they are accessible only by the control software with after some internal processing happened. Nevertheless, these data have to be distinguished from post-processed data where e.g. raw data are converted to interpreted spectra, or orientation mappings.

This post-processing tasks can be performed (on-the-fly, i.e. during acquisition for sure during the microscope session) or afterwards. Post-processing is performed with commercial software or various types and scripts.

Currently, several specializations of NXimage_set and Nspectrum_set are used which store some details of this processing. However, as post- processing tasks can be substantially more advanced and involved it is clear that data artifacts from the measurement and data artifacts generated during post-processing are weakly connected only, maybe exclusively by the fact that a complex numerical post-processing workflow just takes one raw dataset from an NXevent_data_em instance but generates multiple derived data artifacts from this. All these should be described as own application definitions and only weak connections should be made to an instance of NXem. Instances of NXsubentry is one mechanism in NeXus how this can be achieved in the future.
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_data_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 for having an NXevent_data_em instance that contains an NXimage_em or NXspectra_em instance. Thereby, an NXevent_data_em can also be used for just documentation about the specific state of the microscope irrespective whether data have been collected during this time interval.

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 the microscope session and thus all relevant metadata inside the NXinstrument groups are sufficient to understand the session.

So far there exists no standard which a majority of the technology partners and the materials science electron microscopy community have accepted which could be used for a very generic documentation, storage and exchange of electron microscope data. Therefore, it is still a frequent case that specific files have many fields which cannot safely be mapped or interpreted. **Therefore, users are always given the advice to keep the vendor files.** Working however with these vendor files inside specific software, like materials databases, demands for parsers which extract pieces of information from the vendor representation (numerical data and metadata) and map them on a schema with which the database and its associated software tools can work with.

Currently, one would loose immediately track of e.g. provenance and the origin of certain data in NXevent_data_em instances unless really all data are safely and reliably copied over into an instance of the schema. Currently, though, this is sadly effectively prevented in many cases as vendors indeed implemented often sophisticated provenance and commercial software state tracking tools but these are not yet documented covering enough in our opinion so that it is safe to assume all vendor field names are known, logically understood, interpretable, and thus mappable on a common schema using a controlled common vocabulary.

Therefore we encourage user for now to store for each NXimage_set or NXspectra_set instance to supply the so-called source of the data. This can for instance be the name and hashvalue of the original file which was acquired during the microscope session and from which then certain details like numerical data and metadata were copied into an instance of this schema for the purpose of working with the data in e.g. tools offered by research data management (RDM) systems or materials database.

During our work on implementing file format/metadata parsers and developing this application definition, we realized that **several software tools currently do not consistently format critical metadata like time-zone-aware timestamps** when events of data collection or processing happened.

We would like to encourage the community and especially the vendors to work towards a standardization, or at least an open documentation of the way how time-zone-aware time data are collected and stored how and where during a microscope session and how they end up in files and databases with which users interact. This would enable to supplement instances of this schema with specific time data and assure that these time data can be used to reliably contextualize individual datasets and processing steps in materials information systems.

For the reason that these measures have not yet been fully taken, the start_time and end_time is a recommended option. The idea behind these time-zone-aware dates is to identify when the data were collected at the microscope but NOT when they were transcoded by some software tool(s) while storing the data in an instance of this schema. **start_time**: (recommended) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

event_identifier: (recommended) *NX_CHAR* <=

event_type: (recommended) *NX_CHAR* <=

IMAGE_SET: (optional) *NXimage_set* <=

PROCESS: (required) *NXprocess* <=

source: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

mode: (recommended) *NX_CHAR* <=

detector_identifier: (required) *NX_CHAR* <=

PROGRAM: (optional) *NXprogram* <=

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

stack: (recommended) *NXdata* <=

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR* <=

data_counts: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_image_identifier: (required) *NX_UINT* <=

@long_name: (required) *NX_CHAR* <=

axis_y: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_x: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

SPECTRUM_SET: (optional) *NXspectrum_set* <=

PROCESS: (required) *NXprocess* <=

source: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

mode: (recommended) *NX_CHAR* <=

detector_identifier: (required) *NX_CHAR* <=

PROGRAM: (optional) *NXprogram* <=

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

stack: (recommended) *NXdata* <=

data_counts: (required) *NX_NUMBER* (Rank: 3, Dimensions: [n_y, n_x, n_energy]) {units=*NX_UNITLESS*} <=

@long_name: (required) *NX_CHAR* <=

axis_y: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*} <=

@long_name: (required) *NX_CHAR* <=

axis_x: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_LENGTH*} <=

@long_name: (required) *NX_CHAR* <=

axis_energy: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy]) {units=*NX_ENERGY*} <=

@long_name: (required) *NX_CHAR* <=

summary: (recommended) *NXdata* <=

data_counts: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy]) {units=*NX_UNITLESS*} <=

@long_name: (required) *NX_CHAR* <=

axis_energy: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy]) {units=*NX_ENERGY*} <=

@long_name: (required) *NX_CHAR* <=

adf: (optional) *NXimage_set*

inner_half_angle: (recommended) *NX_FLOAT* {units=*NX_ANGLE*}

Annulus inner half angle

outer_half_angle: (recommended) *NX_FLOAT* {units=*NX_ANGLE*}

Annulus outer half angle

PROCESS: (required) *NXprocess* <=

source: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

mode: (recommended) *NX_CHAR* <=

detector_identifier: (required) *NX_CHAR* <=

PROGRAM: (optional) *NXprogram* <=

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

stack: (recommended) *NXdata* <=

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR* <=

data_counts: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_image_identifier: (required) *NX_UINT* <=

@long_name: (required) *NX_CHAR* <=

axis_y: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_x: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

xray: (optional) *NXspectrum_set*

PROCESS: (required) *NXprocess* <=

source: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

mode: (recommended) *NX_CHAR* <=

detector_identifier: (required) *NX_CHAR* <=

PROGRAM: (optional) *NXprogram* <=

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

stack: (required) *NXdata* <=

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR* <=

data_counts: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_y: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_x: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_photon_energy: (required) *NX_NUMBER*

@long_name: (required) *NX_CHAR*

summary: (required) *NXdata* <=

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR* <=

data_counts: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_photon_energy: (required) *NX_NUMBER*

@long_name: (required) *NX_CHAR*

indexing: (optional) *NXprocess* <=

ELEMENTNAME: (required) *NXprocess*

summary: (required) *NXdata*

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR* <=

data_counts: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_y: (required) *NX_NUMBER*

@long_name: (required) *NX_CHAR*

axis_x: (required) *NX_NUMBER*

@long_name: (required) *NX_CHAR*

eels: (optional) *NXspectrum_set*

PROCESS: (required) *NXprocess* <=

source: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

mode: (recommended) *NX_CHAR* <=

detector_identifier: (required) *NX_CHAR* <=

PROGRAM: (optional) *NXprogram* <=

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

stack: (required) *NXdata* <=

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR* <=

data_counts: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_y: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_x: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_energy_loss: (required) *NX_NUMBER*

@long_name: (required) *NX_CHAR*

summary: (required) *NXdata* <=

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR* <=

data_counts: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_energy_loss: (required) *NX_NUMBER*

@long_name: (required) *NX_CHAR*

kikuchi: (optional) *NXimage_set*

PROCESS: (required) *NXprocess* <=

source: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

mode: (recommended) *NX_CHAR* <=

detector_identifier: (required) *NX_CHAR* <=

PROGRAM: (optional) *NXprogram* <=

- - - **program**: (required) *NX_CHAR* <=
      - **@version**: (required) *NX_CHAR* <=
  - **stack**: (required) *NXdata* <=
    - **@signal**: (required) *NX_CHAR* <=
    - **@axes**: (required) *NX_CHAR* <=
    - **@AXISNAME_indices**: (required) *NX_CHAR*
    - **title**: (required) *NX_CHAR* <=
    - **data_counts**: (required) *NX_NUMBER* <=
      - **@long_name**: (required) *NX_CHAR* <=
    - **pattern_identifier**: (required) *NX_UINT*
      - **@long_name**: (required) *NX_CHAR*
    - **axis_y**: (required) *NX_NUMBER* <=
      - **@long_name**: (required) *NX_CHAR* <=
    - **axis_x**: (required) *NX_NUMBER* <=
      - **@long_name**: (required) *NX_CHAR* <=
    - **scan_point_identifier**: (required) *NX_UINT*
- **em_lab**: (optional) *NXinstrument* <=
  - **CHAMBER**: (optional) *NXchamber*
  - **EBEAM_COLUMN**: (optional) *NXebeam_column*
    - **CHAMBER**: (optional) *NXchamber*
    - **electron_source**: (optional) *NXsource* <=
      - **voltage**: (required) *NX_FLOAT* <=
    - **APERTURE_EM**: (optional) *NXaperture_em* <=
      - **value**: (required) *NX_NUMBER* <=
    - **LENS_EM**: (optional) *NXlens_em* <=
      - **voltage**: (recommended) *NX_NUMBER* <=
      - **current**: (recommended) *NX_NUMBER* <=
      - **value**: (recommended) *NX_NUMBER* <=
    - **aberration_correction**: (recommended) *NXcorrector_cs*
      - **applied**: (required) *NX_BOOLEAN* <=
      - **name**: (optional) *NX_CHAR* <=
      - **FABRICATION**: (recommended) *NXfabrication* <=
        - **vendor**: (recommended) *NX_CHAR* <=
        - **model**: (recommended) *NX_CHAR* <=
        - **identifier**: (recommended) *NX_CHAR* <=
      - **ZEMLIN_TABLEAU**: (recommended) *NXprocess* <=

description: (optional) *NX_CHAR* <=

tilt_angle: (recommended) *NX_FLOAT* {units=*NX_ANGLE*} <=

exposure_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

magnification: (optional) *NX_NUMBER* {units=*NX_DIMENSIONLESS*} <=

PROCESS: (optional) *NXprocess* <=

ceos: (optional) *NXaberration_model_ceos* <=

c_1: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

a_1: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

b_2: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

a_2: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

c_3: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

s_3: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

a_3: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

b_4: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

d_4: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

a_4: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (recommended) *NX_FLOAT* {units=*NX_TIME*} <=

c_5: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

- **uncertainty**: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} *<=*
- **uncertainty_model**: (recommended) *NX_CHAR <=*
- **delta_time**: (recommended) *NX_FLOAT* {units=*NX_TIME*} *<=*

s_5: (optional) *NXaberration <=*

- **magnitude**: (required) *NX_FLOAT* {units=*NX_LENGTH*} *<=*
- **angle**: (required) *NX_FLOAT* {units=*NX_ANGLE*} *<=*
- **uncertainty**: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} *<=*
- **uncertainty_model**: (recommended) *NX_CHAR <=*
- **delta_time**: (recommended) *NX_FLOAT* {units=*NX_TIME*} *<=*

r_5: (optional) *NXaberration <=*

- **magnitude**: (required) *NX_FLOAT* {units=*NX_LENGTH*} *<=*
- **angle**: (required) *NX_FLOAT* {units=*NX_ANGLE*} *<=*
- **uncertainty**: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} *<=*
- **uncertainty_model**: (recommended) *NX_CHAR <=*
- **delta_time**: (recommended) *NX_FLOAT* {units=*NX_TIME*} *<=*

a_6: (optional) *NXaberration <=*

- **magnitude**: (required) *NX_FLOAT* {units=*NX_LENGTH*} *<=*
- **angle**: (required) *NX_FLOAT* {units=*NX_ANGLE*} *<=*
- **uncertainty**: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} *<=*
- **uncertainty_model**: (recommended) *NX_CHAR <=*
- **delta_time**: (recommended) *NX_FLOAT* {units=*NX_TIME*} *<=*

nion: (optional) *NXaberration_model_nion* <=

c_1_0: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_1_2_a: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_1_2_b: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_2_1_a: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_2_1_b: (recommended) *NXaberration* <=

    - **magnitude**: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=
    - **uncertainty**: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=
    - **uncertainty_model**: (recommended) *NX_CHAR* <=
    - **delta_time**: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

- **c_2_3_a**: (recommended) *NXaberration* <=
    - **magnitude**: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=
    - **uncertainty**: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=
    - **uncertainty_model**: (recommended) *NX_CHAR* <=
    - **delta_time**: (optional) *NX_FLOAT* {units=*NX_TIME*} <=
- **c_2_3_b**: (recommended) *NXaberration* <=
    - **magnitude**: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=
    - **uncertainty**: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=
    - **uncertainty_model**: (recommended) *NX_CHAR* <=
    - **delta_time**: (optional) *NX_FLOAT* {units=*NX_TIME*} <=
- **c_3_0**: (recommended) *NXaberration* <=
    - **magnitude**: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=
    - **uncertainty**: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=
    - **uncertainty_model**: (recommended) *NX_CHAR* <=
    - **delta_time**: (optional) *NX_FLOAT* {units=*NX_TIME*} <=
- **c_3_2_a**: (recommended) *NXaberration* <=
    - **magnitude**: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_3_2_b: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_3_4_a: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_3_4_b: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_4_1_a: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_4_1_b: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_4_3_a: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_4_3_b: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_4_5_a: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_4_5_b: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_5_0: (recommended) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_5_2_a: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_5_2_b: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_5_4_a: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_5_4_b: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_5_6_a: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

c_5_6_b: (optional) *NXaberration* <=

magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

uncertainty_model: (recommended) *NX_CHAR* <=

delta_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

IBEAM_COLUMN: (optional) *NXibeam_column*

CHAMBER: (optional) *NXchamber*

ion_source: (optional) *NXsource* <=

- **voltage**: (recommended) *NX_FLOAT* <=
- **current**: (recommended) *NX_FLOAT* <=
- **probe**: (recommended) *NXion* <=
  - **charge_state**: (recommended) *NX_INT* <=
  - **isotope_vector**: (recommended) *NX_UINT* <=
  - **name**: (recommended) *NX_CHAR* <=

APERTURE_EM: (optional) *NXaperture_em* <=

- **value**: (recommended) *NX_NUMBER* <=

LENS_EM: (optional) *NXlens_em* <=

- **voltage**: (recommended) *NX_NUMBER* <=
- **current**: (recommended) *NX_NUMBER* <=
- **value**: (recommended) *NX_NUMBER* <=

ebeam_deflector: (optional) *NXscanbox_em*

- **pixel_time**: (recommended) *NX_FLOAT* <=

ibeam_deflector: (optional) *NXscanbox_em*

OPTICAL_SYSTEM_EM: (optional) *NXoptical_system_em*

- **camera_length**: (optional) *NX_NUMBER* <=
- **magnification**: (optional) *NX_NUMBER* <=
- **defocus**: (recommended) *NX_NUMBER* <=
- **semi_convergence_angle**: (recommended) *NX_NUMBER* <=
- **working_distance**: (recommended) *NX_FLOAT* <=
- **beam_current**: (recommended) *NX_FLOAT* <=
- **beam_current_description**: (recommended) *NX_CHAR* <=

DETECTOR: (optional) *NXdetector* <=

PUMP: (optional) *NXpump*

STAGE_LAB: (optional) *NXstage_lab*

- **position**: (recommended) *NX_FLOAT* <=
- **rotation**: (recommended) *NX_FLOAT* <=
- **tilt_1**: (recommended) *NX_FLOAT* <=
- **tilt_2**: (recommended) *NX_FLOAT* <=

USER: (optional) *NXuser* <=

- **name**: (required) *NX_CHAR* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXem/ENTRY-group*
- */NXem/ENTRY/COORDINATE_SYSTEM_SET-group*
- */NXem/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-group*
- */NXem/ENTRY/DATA-group*
- */NXem/ENTRY/definition-field*
- */NXem/ENTRY/em_lab-group*
- */NXem/ENTRY/em_lab/CHAMBER-group*
- */NXem/ENTRY/em_lab/DETECTOR-group*
- */NXem/ENTRY/em_lab/DETECTOR/FABRICATION-group*
- */NXem/ENTRY/em_lab/DETECTOR/FABRICATION/identifier-field*
- */NXem/ENTRY/em_lab/DETECTOR/local_name-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/applied-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/FABRICATION-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/FABRICATION/identifier-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/FABRICATION/model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/FABRICATION/vendor-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/name-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/description-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/exposure_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/magnification-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_1-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_1/angle-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_1/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_1/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_1/uncertainty-field*

- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_1/uncertainty_model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_2-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_2/angle-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_2/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_2/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_2/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_2/uncertainty_model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_3-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_3/angle-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_3/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_3/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_3/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_3/uncertainty_model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_4-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_4/angle-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_4/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_4/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_4/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_4/uncertainty_model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_6-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_6/angle-field*

- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_6/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_6/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_6/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/a_6/uncertainty_model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/b_2-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/b_2/angle-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/b_2/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/b_2/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/b_2/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/b_2/uncertainty_model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/b_4-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/b_4/angle-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/b_4/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/b_4/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/b_4/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/b_4/uncertainty_model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_1-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_1/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_1/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_1/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_1/uncertainty_model-field*

- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_3-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_3/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_3/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_3/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_3/uncertainty_model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_5-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_5/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_5/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_5/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/c_5/uncertainty_model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/d_4-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/d_4/angle-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/d_4/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/d_4/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/d_4/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/d_4/uncertainty_model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/r_5-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/r_5/angle-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/r_5/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/r_5/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/r_5/uncertainty-field*

- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/r_5/uncertainty_model-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/s_3-group
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/s_3/angle-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/s_3/delta_time-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/s_3/magnitude-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/s_3/uncertainty-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/s_3/uncertainty_model-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/s_5-group
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/s_5/angle-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/s_5/delta_time-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/s_5/magnitude-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/s_5/uncertainty-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/ceos/s_5/uncertainty_model-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion-group
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_0-group
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_0/delta_time-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_0/magnitude-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_0/uncertainty-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_0/uncertainty_model-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_2_a-group
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_2_a/delta_time-field
- /NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_2_a/magnitude-field

- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_2_a/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_2_a/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_2_b-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_2_b/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_2_b/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_2_b/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_1_2_b/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_1_a-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_1_a/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_1_a/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_1_a/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_1_a/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_1_b-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_1_b/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_1_b/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_1_b/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_1_b/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_3_a-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_3_a/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_3_a/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_3_a/uncertainty-field*

- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_3_a/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_3_b-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_3_b/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_3_b/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_3_b/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_2_3_b/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_0-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_0/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_0/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_0/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_0/uncertainty_mode field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_2_a-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_2_a/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_2_a/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_2_a/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_2_a/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_2_b-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_2_b/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_2_b/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_2_b/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_2_b/uncertainty_mo field*

- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_4_a-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_4_a/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_4_a/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_4_a/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_4_a/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_4_b-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_4_b/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_4_b/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_4_b/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_3_4_b/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_1_a-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_1_a/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_1_a/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_1_a/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_1_a/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_1_b-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_1_b/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_1_b/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_1_b/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_1_b/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_3_a-group*

- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_3_a/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_3_a/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_3_a/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_3_a/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_3_b-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_3_b/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_3_b/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_3_b/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_3_b/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_5_a-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_5_a/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_5_a/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_5_a/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_5_a/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_5_b-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_5_b/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_5_b/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_5_b/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_4_5_b/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_0-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_0/delta_time-field*

- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_0/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_0/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_0/uncertainty_mode field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_2_a-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_2_a/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_2_a/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_2_a/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_2_a/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_2_b-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_2_b/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_2_b/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_2_b/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_2_b/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_4_a-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_4_a/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_4_a/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_4_a/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_4_a/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_4_b-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_4_b/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_4_b/magnitude-field*

- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_4_b/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_4_b/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_6_a-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_6_a/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_6_a/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_6_a/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_6_a/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_6_b-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_6_b/delta_time-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_6_b/magnitude-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_6_b/uncertainty-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROCESS/nion/c_5_6_b/uncertainty_mo field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/tilt_angle-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/APERTURE_EM-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/APERTURE_EM/description-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/APERTURE_EM/FABRICATION-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/APERTURE_EM/FABRICATION/identifier-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/APERTURE_EM/FABRICATION/model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/APERTURE_EM/FABRICATION/vendor-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/APERTURE_EM/name-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/APERTURE_EM/value-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/CHAMBER-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/electron_source-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/electron_source/emitter_type-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/electron_source/FABRICATION-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/electron_source/FABRICATION/identifier-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/electron_source/FABRICATION/model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/electron_source/FABRICATION/vendor-field*

- */NXem/ENTRY/em_lab/EBEAM_COLUMN/electron_source/name-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/electron_source/voltage-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/FABRICATION-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/FABRICATION/identifier-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/FABRICATION/model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/FABRICATION/vendor-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/LENS_EM-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/LENS_EM/current-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/LENS_EM/FABRICATION-group*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/LENS_EM/FABRICATION/identifier-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/LENS_EM/FABRICATION/model-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/LENS_EM/FABRICATION/vendor-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/LENS_EM/value-field*
- */NXem/ENTRY/em_lab/EBEAM_COLUMN/LENS_EM/voltage-field*
- */NXem/ENTRY/em_lab/EBEAM_DEFLECTOR-group*
- */NXem/ENTRY/em_lab/EBEAM_DEFLECTOR/FABRICATION-group*
- */NXem/ENTRY/em_lab/EBEAM_DEFLECTOR/FABRICATION/identifier-field*
- */NXem/ENTRY/em_lab/EBEAM_DEFLECTOR/FABRICATION/model-field*
- */NXem/ENTRY/em_lab/EBEAM_DEFLECTOR/FABRICATION/vendor-field*
- */NXem/ENTRY/em_lab/EBEAM_DEFLECTOR/pixel_time-field*
- */NXem/ENTRY/em_lab/FABRICATION-group*
- */NXem/ENTRY/em_lab/FABRICATION/capabilities-field*
- */NXem/ENTRY/em_lab/FABRICATION/identifier-field*
- */NXem/ENTRY/em_lab/FABRICATION/model-field*
- */NXem/ENTRY/em_lab/FABRICATION/vendor-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN-group*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/APERTURE_EM-group*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/APERTURE_EM/FABRICATION-group*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/APERTURE_EM/FABRICATION/identifier-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/APERTURE_EM/FABRICATION/model-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/APERTURE_EM/FABRICATION/vendor-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/APERTURE_EM/value-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/CHAMBER-group*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/FABRICATION-group*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/FABRICATION/identifier-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/FABRICATION/model-field*

- */NXem/ENTRY/em_lab/IBEAM_COLUMN/FABRICATION/vendor-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/ion_source-group*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/ion_source/current-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/ion_source/emitter_type-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/ion_source/FABRICATION-group*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/ion_source/FABRICATION/identifier-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/ion_source/FABRICATION/model-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/ion_source/FABRICATION/vendor-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/ion_source/name-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/ion_source/probe-group*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/ion_source/probe/charge_state-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/ion_source/probe/isotope_vector-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/ion_source/probe/name-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/ion_source/voltage-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/LENS_EM-group*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/LENS_EM/current-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/LENS_EM/FABRICATION-group*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/LENS_EM/FABRICATION/identifier-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/LENS_EM/value-field*
- */NXem/ENTRY/em_lab/IBEAM_COLUMN/LENS_EM/voltage-field*
- */NXem/ENTRY/em_lab/IBEAM_DEFLECTOR-group*
- */NXem/ENTRY/em_lab/IBEAM_DEFLECTOR/FABRICATION-group*
- */NXem/ENTRY/em_lab/IBEAM_DEFLECTOR/FABRICATION/identifier-field*
- */NXem/ENTRY/em_lab/IBEAM_DEFLECTOR/FABRICATION/model-field*
- */NXem/ENTRY/em_lab/IBEAM_DEFLECTOR/FABRICATION/vendor-field*
- */NXem/ENTRY/em_lab/instrument_name-field*
- */NXem/ENTRY/em_lab/location-field*
- */NXem/ENTRY/em_lab/OPTICAL_SYSTEM_EM-group*
- */NXem/ENTRY/em_lab/OPTICAL_SYSTEM_EM/beam_current-field*
- */NXem/ENTRY/em_lab/OPTICAL_SYSTEM_EM/beam_current_description-field*
- */NXem/ENTRY/em_lab/OPTICAL_SYSTEM_EM/camera_length-field*
- */NXem/ENTRY/em_lab/OPTICAL_SYSTEM_EM/defocus-field*
- */NXem/ENTRY/em_lab/OPTICAL_SYSTEM_EM/magnification-field*
- */NXem/ENTRY/em_lab/OPTICAL_SYSTEM_EM/semi_convergence_angle-field*
- */NXem/ENTRY/em_lab/OPTICAL_SYSTEM_EM/working_distance-field*
- */NXem/ENTRY/em_lab/PUMP-group*

- /NXem/ENTRY/em_lab/stage_lab-group
- /NXem/ENTRY/em_lab/stage_lab/description-field
- /NXem/ENTRY/em_lab/stage_lab/design-field
- /NXem/ENTRY/em_lab/stage_lab/FABRICATION-group
- /NXem/ENTRY/em_lab/stage_lab/FABRICATION/capabilities-field
- /NXem/ENTRY/em_lab/stage_lab/FABRICATION/identifier-field
- /NXem/ENTRY/em_lab/stage_lab/FABRICATION/model-field
- /NXem/ENTRY/em_lab/stage_lab/FABRICATION/vendor-field
- /NXem/ENTRY/em_lab/stage_lab/name-field
- /NXem/ENTRY/em_lab/stage_lab/position-field
- /NXem/ENTRY/em_lab/stage_lab/rotation-field
- /NXem/ENTRY/em_lab/stage_lab/tilt_1-field
- /NXem/ENTRY/em_lab/stage_lab/tilt_2-field
- /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/adf-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/inner_half_angle-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/outer_half_angle-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/PROCESS-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/PROCESS/detector_identifier-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/PROCESS/mode-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/PROCESS/PROGRAM-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/PROCESS/PROGRAM/program-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/PROCESS/PROGRAM/program@version-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/PROCESS/source-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/PROCESS/source@version-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/stack-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/stack/axis_image_identifier-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/stack/axis_image_identifier@long_name-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/stack/axis_x-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/stack/axis_x@long_name-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/stack/axis_y-field

- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/stack/axis_y@long_name-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/stack/data_counts-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/stack/data_counts@long_name-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/stack/title-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/stack@axes-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/stack@AXISNAME_indices-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/adf/stack@signal-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/PROCESS-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/PROCESS/detector_identifier-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/PROCESS/mode-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/PROCESS/PROGRAM-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/PROCESS/PROGRAM/program-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/PROCESS/PROGRAM/program@version-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/PROCESS/source-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/PROCESS/source@version-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/stack-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/stack/axis_energy_loss-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/stack/axis_energy_loss@long_name-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/stack/axis_x-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/stack/axis_x@long_name-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/stack/axis_y-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/stack/axis_y@long_name-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/stack/data_counts-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/stack/data_counts@long_name-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/stack/title-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/stack@axes-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/stack@AXISNAME_indices-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/stack@signal-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/summary-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/summary/axis_energy_loss-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/summary/axis_energy_loss@long_name-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/summary/data_counts-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/summary/data_counts@long_name-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/summary/title-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/summary@axes-attribute

- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/summary@AXISNAME_indices-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/eels/summary@signal-attribute
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/CHAMBER-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/DETECTOR-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/applied-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/FABRICATION-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/FABRICATION/identifier-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/FABRICATION/model-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/FABRICATION/vendor-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/name-field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU-group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/descri field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/exposu field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/magni field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field

- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*

- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*

- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field

- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*

- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*

- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*

- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field
- /NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field

- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*

- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*

- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/PROC field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/aberration_correction/ZEMLIN_TABLEAU/tilt_an field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/APERTURE_EM-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/APERTURE_EM/value-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/CHAMBER-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/electron_source-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/electron_source/voltage-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/LENS_EM-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/LENS_EM/current-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/LENS_EM/value-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/EBEAM_COLUMN/LENS_EM/voltage-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/ebeam_deflector-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/ebeam_deflector/pixel_time-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN/APERTURE_EM-group*

- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN/APERTURE_EM/value-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN/CHAMBER-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN/ion_source-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN/ion_source/current-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN/ion_source/probe-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN/ion_source/probe/charge_state-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN/ion_source/probe/isotope_vector-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN/ion_source/probe/name-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN/ion_source/voltage-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN/LENS_EM-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN/LENS_EM/current-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN/LENS_EM/value-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/IBEAM_COLUMN/LENS_EM/voltage-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/ibeam_deflector-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/OPTICAL_SYSTEM_EM-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/OPTICAL_SYSTEM_EM/beam_current-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/OPTICAL_SYSTEM_EM/beam_current_description-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/OPTICAL_SYSTEM_EM/camera_length-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/OPTICAL_SYSTEM_EM/defocus-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/OPTICAL_SYSTEM_EM/magnification-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/OPTICAL_SYSTEM_EM/semi_convergence_angle-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/OPTICAL_SYSTEM_EM/working_distance-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/PUMP-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/STAGE_LAB-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/STAGE_LAB/position-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/STAGE_LAB/rotation-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/STAGE_LAB/tilt_1-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/em_lab/STAGE_LAB/tilt_2-field*
- */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/IMAGE_SET-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/PROCESS-group*

- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/PROCESS/detector_identifier-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/PROCESS/mode-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/PROCESS/PROGRAM-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/PROCESS/PROGRAM/program-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/PROCESS/PROGRAM/program@version-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/PROCESS/source-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/PROCESS/source@version-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/stack-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/stack/axis_image_identifier-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/stack/axis_image_identifier@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/stack/axis_x-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/stack/axis_x@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/stack/axis_y-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/stack/axis_y@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/stack/data_counts-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/stack/data_counts@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/stack/title-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/stack@axes-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/stack@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/IMAGE_SET/stack@signal-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/PROCESS-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/PROCESS/detector_identifier-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/PROCESS/mode-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/PROCESS/PROGRAM-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/PROCESS/PROGRAM/program-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/PROCESS/PROGRAM/program@version-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/PROCESS/source-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/PROCESS/source@version-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/stack-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/stack/axis_x-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/stack/axis_x@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/stack/axis_y-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/stack/axis_y@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/stack/data_counts-field*

- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/stack/data_counts@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/stack/pattern_identifier-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/stack/pattern_identifier@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/stack/scan_point_identifier-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/stack/title-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/stack@axes-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/stack@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/kikuchi/stack@signal-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/PROCESS-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/PROCESS/detector_identifier-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/PROCESS/mode-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/PROCESS/PROGRAM-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/PROCESS/PROGRAM/program-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/PROCESS/PROGRAM/program@version-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/PROCESS/source-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/PROCESS/source@version-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/stack-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/stack/axis_energy-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/stack/axis_energy@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/stack/axis_x-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/stack/axis_x@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/stack/axis_y-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/stack/axis_y@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/stack/data_counts-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/stack/data_counts@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/summary-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/summary/axis_energy-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/summary/axis_energy@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/summary/data_counts-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/SPECTRUM_SET/summary/data_counts@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/start_time-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/USER-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/USER/name-field*

- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/indexing-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/indexing/ELEMENTNAME-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/indexing/ELEMENTNAME/summary-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/indexing/ELEMENTNAME/summary/axis_x-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/indexing/ELEMENTNAME/summary/axis_x@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/indexing/ELEMENTNAME/summary/axis_y-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/indexing/ELEMENTNAME/summary/axis_y@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/indexing/ELEMENTNAME/summary/data_counts-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/indexing/ELEMENTNAME/summary/data_counts@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/indexing/ELEMENTNAME/summary/title-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/indexing/ELEMENTNAME/summary@axes-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/indexing/ELEMENTNAME/summary@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/indexing/ELEMENTNAME/summary@signal-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/PROCESS-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/PROCESS/detector_identifier-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/PROCESS/mode-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/PROCESS/PROGRAM-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/PROCESS/PROGRAM/program-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/PROCESS/PROGRAM/program@version-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/PROCESS/source-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/PROCESS/source@version-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/stack-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/stack/axis_photon_energy-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/stack/axis_photon_energy@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/stack/axis_x-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/stack/axis_x@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/stack/axis_y-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/stack/axis_y@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/stack/data_counts-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/stack/data_counts@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/stack/title-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/stack@axes-attribute*

- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/stack@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/stack@signal-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/summary-group*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/summary/axis_photon_energy-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/summary/axis_photon_energy@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/summary/data_counts-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/summary/data_counts@long_name-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/summary/title-field*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/summary@axes-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/summary@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/EVENT_DATA_EM/xray/summary@signal-attribute*
- */NXem/ENTRY/MONITOR-group*
- */NXem/ENTRY/PROGRAM-group*
- */NXem/ENTRY/PROGRAM/program-field*
- */NXem/ENTRY/PROGRAM/program@version-attribute*
- */NXem/ENTRY/sample-group*
- */NXem/ENTRY/sample/atom_types-field*
- */NXem/ENTRY/sample/density-field*
- */NXem/ENTRY/sample/description-field*
- */NXem/ENTRY/sample/method-field*
- */NXem/ENTRY/sample/name-field*
- */NXem/ENTRY/sample/preparation_date-field*
- */NXem/ENTRY/sample/sample_history-field*
- */NXem/ENTRY/sample/short_title-field*
- */NXem/ENTRY/sample/thickness-field*
- */NXem/ENTRY/start_time-field*
- */NXem/ENTRY/thumbnail-group*
- */NXem/ENTRY/thumbnail@type-attribute*
- */NXem/ENTRY/USER-group*
- */NXem/ENTRY/USER/address-field*
- */NXem/ENTRY/USER/affiliation-field*
- */NXem/ENTRY/USER/email-field*
- */NXem/ENTRY/USER/name-field*
- */NXem/ENTRY/USER/orcid-field*
- */NXem/ENTRY/USER/orcid_platform-field*
- */NXem/ENTRY/USER/role-field*

- */NXem/ENTRY/USER/social_media_name-field*
- */NXem/ENTRY/USER/social_media_platform-field*
- */NXem/ENTRY/USER/telephone_number-field*
- */NXem/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXem.nxdl.xml

NXem_ebsd

Status:

application definition, extends *NXobject*

Description:

Application definition for collecting and indexing Kikuchi pattern into orientation maps.

This NXem_ebsd application is a proposal how to represent data, metadata, and connections between these for the research field of electron microscopy. More specifically, exemplified here for electron backscatter diffraction (EBSD). The application definition solves two key documentation issues which are missing so far to document provenance of data and metadata in the field of EBSD. The application definition can be an example that is relevant for related workflows in orientation microscopy.

Firstly, an instance of NXem_ebsd (such as a NeXus/HDF5 file which is formatted according to the NXem_ebsd application definition) stores the connection between the microscope session and the key datasets which are considered typically results of the various processing steps involved when working with EBSD data.

Different groups in this application definition make connections to data artifacts which were collected when working with electron microscopes via the NXem partner application definition. Using a file which stores information according to the NXem application definition has the benefit that it connects the sample, references to the sample processing, the user operating the microscope, details about the microscope session, and details about the acquistion and eventual indexing of Kikuchi pattern, associated overview images, like secondary electron or backscattered electron images of the region-of-interest probed and many more pieces of information.

Secondly, this NXem_ebsd application definition connects and stores the conventions and reference frames which were used and are the key to mathematically correctly interpret every EBSD result. Otherwise, results would be ripped out of their context, as it is the situation with many traditional studies where EBSD data were indexed on-the-fly and shared with the community only via sharing the results file with some technology-partner-specific file but leaving important conventions out or relying on the assumptions that colleagues know these even though multiple definitions are possible.

This application definition covers experiments with one-, two-dimensional, and so-called three-dimensional EBSD datasets. The third dimension is either time (in the case of quasi in-situ experiments) or space (in the case of serial- sectioning) methods where a combination of mechanical or ion milling is used repetitively to measure the same region-of-interest at different depth increments. Material removal can be achieved with electron or ion polishing, using manual steps or using automated equipment like a robot system.

Three-dimensional experiments require to follow a sequence of specimen, surface preparation, and data collection steps. By nature these methods are destructive in that they either require the removal of the previously measured material region or that the sample surface can degrade due to e.g. contamination or other electron-matter interaction.

For three-dimensional EBSD, multiple two-dimensional EBSD orientation mappings are combined into one reconstructed stack. That is serial-sectioning is mainly a computational workflow. Users collect data for each serial sectioning step via an experiment. This assures that data for associated microscope sessions and steps of data processing stay connected and contextualized.

Eventual tomography methods also use such a workflow because first diffraction images are collected (e.g. with X-ray) and then these imagres are indexed and computed into a 3D orientation mapping. The here proposed NXem_ebsd application definition contains conceptual ideas how this splitting between measurement and post-processing can be granularized also for such X-ray-based techniques, whether it be 3DXRD or HEDM.

Symbols:

n_op: Number of arguments per orientation for given parameterization.

n_sc: Number of scan points.

n_z: Number of pixel along the slowest changing dimension for a rediscretized, i.e. standardized default orientation mapping.

n_y: Number of pixel along slow changing dimension for a rediscretized i.e. standardized default orientation mapping.

n_x: Number of pixel along fast changing dimension for a rediscretized i.e. standardized default orientation mapping.

Groups cited:
NXcg_geodesic_mesh, *NXdata*, *NXem_ebsd_conventions*, *NXem_ebsd_crystal_structure_model*, *NXentry*, *NXimage_set_em_kikuchi*, *NXprocess*, *NXprogram*, *NXtransformations*, *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_ebsd`

workflow_identifier: (required) *NX_CHAR*

Ideally, a (globally) unique persistent identifier for referring to this workflow.

The identifier is usually defined/issued by the facility, laboratory, or the principle investigator. The identifier enables to link workflows/experiments to e.g. proposals.

workflow_description: (optional) *NX_CHAR*

Free-text description about the workflow.

Users are strongly advised to detail the sample history in the respective field and fill rather as completely as possible the fields of the application definition behind instead of filling in these details into the experiment_description free-text description field.

start_time: (recommended) *NX_DATE_TIME <=*

ISO 8601 time code with local time zone offset to UTC information included when the processing of the workflow started. If the application demands that time codes in this section of the application definition should only be used for specifying when the workflow was executed - 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 workflow.

end_time: (recommended) *NX_DATE_TIME* <=

ISO 8601 time code with local time zone offset to UTC included when the processing of the workflow ended.

PROGRAM: (required) *NXprogram*

Program which was used for creating the file instance which is formatted according to the NXem_ebsd application definition. **program**: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

USER: (optional) *NXuser* <=

Contact information and eventually details of at least one person involved in performing the workflow. 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: (recommended) *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. If this field is field the specific service should also be written in orcid_platform

orcid_platform: (recommended) *NX_CHAR* <=

Name of the OrcID or ResearcherID where the account under orcid is registered.

telephone_number: (optional) *NX_CHAR* <=

(Business) (tele)phone number of the user at the point in time when the experiment was performed.

role: (recommended) *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.

conventions: (required) *NXem_ebsd_conventions*

rotation_conventions: (required) *NXprocess* <=

three_dimensional_rotation_handedness: (required) *NX_CHAR* <=

rotation_convention: (required) *NX_CHAR* <=

euler_angle_convention: (required) *NX_CHAR* <=

axis_angle_convention: (required) *NX_CHAR* <=

orientation_parameterization_sign_convention: (required) *NX_CHAR* <=

processing_reference_frame: (required) *NXprocess* <=

reference_frame_type: (required) *NX_CHAR* <=

xaxis_direction: (required) *NX_CHAR* <=

xaxis_alias: (required) *NX_CHAR* <=

yaxis_direction: (required) *NX_CHAR* <=

yaxis_alias: (required) *NX_CHAR* <=

zaxis_direction: (required) *NX_CHAR* <=

zaxis_alias: (required) *NX_CHAR* <=

origin: (required) *NX_CHAR* <=

sample_reference_frame: (required) *NXprocess* <=

reference_frame_type: (required) *NX_CHAR* <=

xaxis_direction: (required) *NX_CHAR* <=

yaxis_direction: (required) *NX_CHAR* <=

zaxis_direction: (required) *NX_CHAR* <=

origin: (required) *NX_CHAR* <=

detector_reference_frame: (required) *NXprocess* <=

reference_frame_type: (required) *NX_CHAR* <=

xaxis_direction: (required) *NX_CHAR* <=

yaxis_direction: (required) *NX_CHAR* <=

zaxis_direction: (required) *NX_CHAR* <=

origin: (required) *NX_CHAR* <=

gnomonic_projection_reference_frame: (required) *NXprocess* <=

reference_frame_type: (required) *NX_CHAR* <=

xaxis_direction: (required) *NX_CHAR* <=

yaxis_direction: (required) *NX_CHAR* <=

zaxis_direction: (required) *NX_CHAR* <=

origin: (required) *NX_CHAR <=*

pattern_centre: (required) *NXprocess <=*

xaxis_boundary_convention: (required) *NX_CHAR <=*

xaxis_normalization_direction: (required) *NX_CHAR <=*

yaxis_boundary_convention: (required) *NX_CHAR <=*

yaxis_normalization_direction: (required) *NX_CHAR <=*

simulation: (recommended) *NXprocess <=*

Details about simulations for Kikuchi pattern using kinematic or dynamic diffraction theory. Usually, the output of such computer simulations are spherical Kikuchi images which only when projected or observed in some region-of-interest will represent a set of rectangular Kikuchi pattern with the same rectangular shape and image size.

Therefore, these pattern should be stored. The spherical diffraction pattern can be stored as a set of triangulated geodesic meshes. The rectangular patterns should be stored as NXimage_set_em_kikuchi stack.

Do not store pattern in the simulation group if they have been measured are not simulated. **sequence_index**: (required) *NX_POSINT <=*

PROGRAM: (optional) *NXprogram*

program: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

CG_GEODESIC_MESH: (optional) *NXcg_geodesic_mesh*

IMAGE_SET_EM_KIKUCHI: (required) *NXimage_set_em_kikuchi*

stack: (required) *NXdata <=*

@signal: (required) *NX_CHAR <=*

@axes: (required) *NX_CHAR <=*

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR <=*

data_counts: (required) *NX_NUMBER <=*

@long_name: (required) *NX_CHAR <=*

pattern_identifier: (required) *NX_UINT <=*

@long_name: (required) *NX_CHAR <=*

axis_y: (required) *NX_NUMBER <=*

@long_name: (required) *NX_CHAR <=*

axis_x: (required) *NX_NUMBER <=*

@long_name: (required) *NX_CHAR <=*

experiment: (optional) *NXprocess <=*

The experiment group captures relevant details about the conditions of and the tools used for collecting the Kikuchi diffraction pattern.

The most frequently collected EBSD data are captured as rectangular ROIs composed from square or hexagonally-shaped pixels. Substantially less frequently, because such experiments are more costly and technically demanding, correlated experiments are performed.

One important class of such correlated experiments are the so-called (quasi) in-situ experiments. Here the same or nearly the same ROI is analyzed via a cycles of thermo-mechanical treatment, sample preparation, measurement, on-the-fly-indexing. Phenomena investigated like this are recrystallization, strain accumulation, material damage. Post-processing is required to correlate and reidentify eventual features or local ROIs across several orientation maps.

Another important class of correlated experiments are the so-called serial-sectioning experiments. Here the same sample is repetitively measured and polished to create a stack of orientation data which can be reconstructed to a three-dimensional volume ROI. **time**: (optional) *NX_NUMBER* {units=*NX_TIME*}

> Physical time since the beginning of a timestamp that is required to be same for all experiments in the set. The purpose of this marker is to identify how all experiments in the set have have to be arranged sequentially based on the time elapsed. The time is relevant to sort e.g. experiments of consecutive quasi in-situ experiments where a measurement was e.g. taken after 0 minutes of annealing, 30 minutes, 6 hours, or 24 hours of annealing.

TRANSFORMATIONS: (optional) *NXtransformations*

> Transformation which details where the region-of-interest described under indexing is located in absolute coordinates and rotation with respect to which coordinate system.

calibration: (recommended) *NXprocess*

> The EBSD system, including components like the electron gun, pole-piece, stage tilting, EBSD detector, and the gnomonic projection have to be calibrated to achieve reliable results. Specifically, the gnomonic projection has to be calibrated.
>
> In most practical cases, especially in engineering, there is a substantially larger number of sessions where such a calibrated system is used assuming that somebody has properly calibrated the system rather than that the user actively recalibrates it or is even allowed to do so. Especially the projection geometry has to calibrated which is usually achieved with measuring silicon, quartz or standards, and comparing against simulated diffraction pattern.
>
> In the first case, the user assumes that the principle geometry of the hardware components and the settings in the control and EBSD pattern acquisition software are calibrated. Consequently, users pick from an existent library of phase candidates. One example are the CRY or CIF files of the classical HKL/Channel 5/Flamenco software products. Each entry of the library of such phase candidates in this NeXus proposal is represented by one NXem_ebsd_crystal_structure_model base class. For each phase an instance of this base class is to be used to store crystallographic and simulation-relevant data.
>
> Indexing is a data processing step performed after/during the beam scans the specimen (depends on configuration). Users load the specimen, and first collect a coarse image of the surface. Next, an approximate value for the calibrated working distance is chosen and the stage tilted. Users then may configure the microscope for collecting higher quality data and push in the EBSD detector. Subsequently, they fine tune the illumination and aberration settings and select one or multiple ROIs to machine off. The on-the-fly indexing

parameter are defined and usually the automated measurement queue started.

Nowadays, this is usually an automated/unsupervised process. The pattern collection runs during the allocated session time slot which the user has booked ends or when the queue finishes prematurely. Kikuchi pattern surplus eventually multi-modal detector signals are collected and usually indexed on-the-fly. The Kikuchi patterns may or not be deleted directly after a solution was found (on-the-fly) so Kikuchi pattern are not always stored.

Results files are in many labs afterwards copied automatically for archival purposes to certain storage locations. The result of such an EBSD measurement/experiment is a set of usually proprietary or open files from technology partners (microscope and EBSD detector manufacturers).

In the second case, the system is being calibrated during the session using standards (silicon, quartz, or other common specimens). There is usually one person in each lab responsible for doing such calibrations. Important is that often this person or technician(s) are also in charge of configuring the graphical user interface and software with which most users control and perform their analyses. For EBSD this has key implications because, taking TSL OIM/EDAX as an example, the conventions how orientations are stored is affected by how reference frames are set up and this setup is made at the level of the GUI software. Unfortunately, these pieces of information are not necessarily stored in the results files. In effect, key conventions become disconnected from the data so it remains the users personal obligation to remember these settings, write them down in the lab notebook, or these metadata get lost. All these issues are a motivation and problem which NXem_ebsd solves.
sequence_index: (required) *NX_POSINT <=*

origin: (required) *NX_CHAR*

A link/cross reference to an existent instance of NXem_ebsd with ideally an associated instance of NXem detailed under measurement which informs about the calibration procedures. **@version**: (required) *NX_CHAR*

Commit identifying this resource.

path: (required) *NX_CHAR*

Path which resolves which specific NXimage_set_em_kikuchi instance was used as the raw data to the EBSD data (post)-processing workflow when performing the calibration.

acquisition: (recommended) *NXprocess*

Relevant result of the session at the microscope for this experiment which enables to connect the measurement of the Kikuchi pattern and their processing into orientation microscopy maps. **sequence_index**: (required) *NX_POSINT <=*

origin: (required) *NX_CHAR*

Name or link to an existent instance of an EBSD raw dataset ideally as an instance of an NXem application definition which has at least one NXimage_set_em_kikuchi instance i.e. one stack of Kikuchi pattern. The path to this instance in the origin has to be specified under path.

When NXem is not used or the aim is to rather explore first how community-specific files with EBSD data, such as ANG, CPR, or HDF5-

based formats can be parsed from, inject here the name of that file.

The em_om parser will currently not interpret the majority of the many system- and technique-specific metadata which come with the files from e.g. technology partners. This is because the current culture in the EBSD community is that many of the metadata fields are neither in all cases fully documented nor use a standardized vocabulary although many people understand terms from different implementations and how these metadata can likely be compared to one another.

In addition, it is common practice in the research field of EBSD that users transcode their raw data into other (often text-based or HDF5) files with custom formatting to realize an information transfer between specific software tools including commercial software from technology partner, custom scripts in Matlab using tools like MTex, or Python scripting with tools like hyperspy, pyxem, orix, diffsims, kikuchipy, or EBSD data stack alignment tools like DREAM.3D. We have opted that in the first iteration this implementation of a RDMS-agnostic FAIR data schema for EBSD that we discard these metadata because these ad hoc file formats are not designed to communicate also specifically and most importantly the eventually different context of the metadata. Another reason for this choice was also to emphasize that in fact such challenges do exist in the community and thus pointing them out may support the discussion to arrive at eventually more complete solutions. As developing these solutions should not be our authority and necessarily demands feedback from the technology partners, we have opted for this intermediate approach to stimulate discussion. **@version**: (required) *NX_CHAR*

Commit or e.g. at least SHA256 checksum identifying this resource.

path: (required) *NX_CHAR*

Path which resolves which specific NXimage_set_em_kikuchi instance was used as the raw data to this EBSD data (post)-processing workflow.

indexing: (recommended) *NXprocess*

OIM, orientation imaging microscopy. Post-processing of the Kikuchi patterns to obtain orientation per phase model and scan point. Fundamentally different algorithms can be used to index EBSD/EBSP pattern.

Common is that pattern indexing is a computational step of comparing simulated with measured diffraction pattern. Quality descriptors are defined based on which an indexing algorithm yields a quantitative measure of how similar measured and assumed/simulated pattern are, and thus if no, one, or multiple so-called solutions were found.

Assumed or simulated pattern use kinematical or dynamical electron diffraction theory. Hough transform (which is essentially a discretized Radon transform, for details see e.g A short introduction to the Radon and Hough transforms and how they relate by M. van Ginkel et al.). Recently, dictionary-based indexing methods are increasingly becoming used partly driven by the move to use artificial intelligence algorithms.

An inspection of publicly available EBSD datasets with an open-source license which are available on Zenodo was performed prior to implementing of the associated em_om parser for NXem_ebsd. This analysis revealed that EBSD data are in most cases stored in two ways: Case one was via a file in formats from technology partners. Examples are binary formats like OSC, H5OINA, OIP, EBSP, and many derived text-based formats like CPR, CRC, ANG, CTF, HKL and more. Recently, there is trend towards using HDF5-based formats.

These files contain some result and metadata to the numerical steps and the computational workflow which was performed to index Kikuchi pattern on-the-fly. Examples of metadata include scan point positions, indexing solutions per scan point, some quality descriptors for the solutions, as well as crystal structure and phase metadata.

Case two were raw pattern in some custom format, often text-based with some but in general no conclusive and interoperable representation of all relevant metadata. Often it remains unclear what individual fields and data arrays of these fields resolve and/or mean conceptually. For some fields, publications were referred to. However, software tools change over time and thus which specific data end in a file and which specific conceptual information is behind these data can change with software versions.

Other cases were storing results of custom post-processing steps and associated Kikuchi pattern. Testing of advanced indexing, pseudo-symmetry resolving methods, i.e. any sort of prototyping or alternative indexing strategies so far seem to require some flexibility for implementing rapid prototypic capabilities. The drawback of this is that such results come formatted on a case-by-case basis and are thus not interoperable.

Therefore, we first need to collect how these files have been generated and which metadata in these files (or database entries) represent which pieces of information conceptually. Ideally, one would do so by creating a complete set of information in e.g. an NXem application definition, such as a log of timestamped events and processing steps, metadata and data. Eventually even interactions with the graphical user interface of commercial software during the microscope session should be stored and become a part of the application definition.

Such a set of pieces of information could then be used via reading directly for the NXem application definition. However, in most cases such a data representation is not available yet. **sequence_index**: (required) *NX_POSINT* *<=*

method: (required) *NX_CHAR*

Principal algorithm used for indexing.

Any of these values:

- `undefined`
- `hough_transform`
- `dictionary`
- `radon_transform`
- `other`

status: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_sc]) {units=*NX_UNITLESS*}

Which return value did the indexing algorithm yield for each scan point. Practically useful is to use an uint8 mask.

- 0 - Not analyzed
- 1 - Too high angular deviation
- 2 - No solution
- 100 - Success
- 255 - Unexpected errors

n_phases_per_scan_point: (recommended) *NX_UINT* (Rank: 1, Dimensions: [n_sc]) {units=*NX_UNITLESS*}

How many phases i.e. crystal structure models were used to index each scan point if any? Let's assume an example to explain how this field should be used: In the simplest case users collected one pattern for each scan point and have indexed using one phase, i.e. one instance of an NXem_ebsd_crystal_structure_model.

In another example users may have skipped some scan points (not indexed) them at all) and/or used differing numbers of phases for different scan points.

The cumulated of this array decodes how phase_identifier and phase_matching arrays have to be interpreted. In the simplest case (one pattern per scan point, and all scan points indexed using that same single phase model), phase_identifier has as many entries as scan points and phase_matching has also as many entries as scan points.

phase_identifier: (recommended) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

The array n_phases_per_scan_point details how the phase_identifier and the phase_matching arrays have to be interpreted.

For the example with a single phase phase_identifier has trivial values either 0 (no solution) or 1 (solution matching sufficiently significant with the model for phase 1).

When there are multiple phases, it is possible (although not frequently needed) that a pattern matches eventually (not equally well) sufficiently significant with multiple pattern. This can especially happen in cases of pseudosymmetry and more frequently with an improperly calibrated system or false or inaccurate phase models e.g. (ferrite, austenite). Having such field is especially relevant for recent machine learning or dictionary based indexing schemes because in combination with phase_matching these fields communicate the results in a model-agnostic way.

Depending on the n_phases_per_scan_point value phase_identifier and phase_matching arrays represent a collection of concatenated tuples, which are organized in sequence: The solutions for the 0-th scan point, the 1-th scan point, the n_sc - 1 th scan point and omitting tuples for those scan points with no phases according to n_phases_per_scan_point

phase_matching: (recommended) *NX_NUMBER* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

One-dimensional array, pattern by pattern labelling the solutions found. The array n_phases_per_scan_point has to be specified because it details how the phase_identifier and the phase_matching arrays have to be interpreted. See documentation of phase_identifier for further details.

phase_matching_descriptor: (recommended) *NX_CHAR*

Phase_matching is a descriptor for how well the solution matches or not. Examples can be confidence index (ci), mean angular deviation (mad), some AI-based matching probability (other), i.e. the details are implementation-specific.

Any of these values: `undefined` | `ci` | `mad` | `other`

orientation_parameterization: (recommended) *NX_CHAR*

How are orientations parameterized? Inspect euler_angle_convention in case of using euler to clarify the sequence of rotations assumed.

Any of these values:

- `euler`
- `axis_angle`
- `rodrigues`
- `quaternion`
- `homochoric`

orientation: (recommended) *NX_NUMBER* (Rank: 2, Dimensions: [i, n_op]) {units=*NX_ANY*}

Matrix of parameterized orientations identified. The slow dimension iterates of the individual solutions as defined by n_phases_per_scan_point. Values for phases without a solution should be correctly identified as IEEE NaN.

scan_point_positions: (recommended) *NX_NUMBER* (Rank: 2, Dimensions: [n_sc, 2]) {units=*NX_LENGTH*}

Matrix of calibrated centre positions of each scan point in the sample surface reference system.

hit_rate: (optional) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

Fraction of successfully indexed pattern of the set averaged over entire set.

on_the_fly_indexing: (optional) *NXprocess*

Therefore, the on_the_fly_indexing group stores which source_file contains the results of the on-the-fly indexing. For commercial systems these files can be e.g. ANG, CPR/CRC, H5OINA, OSC. It is possible that the file or database entry which is referred to under origin is the same as the one under a given acquisition/origin in one of the experiment groups. This is because some commercial file formats make no clear distinction between which metadata are acquisition and/or indexing metadata. **origin**: (required) *NX_CHAR*

Name of the file from which data relevant for creating default plots were taken in the case that the data in the experiment group were indexed on-the-fly. **@version**: (required) *NX_CHAR*

Hash of that file.

path: (required) *NX_CHAR*

TBD, path which resolves which specific NXimage_set_em_kikuchi instance was used as the raw data to this EBSD data (post)-processing workflow when performing the calibration.

PROGRAM: (required) *NXprogram*

Commercial program which was used to index the EBSD data incrementally after they have been captured and while the microscope was capturing (on-the-fly). This is the usual production workflow how EBSD data are collected in materials engineering, in industry, and academia. **program**: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

background_correction: (optional) *NXprocess*

Details about the background correction applied to each Kikuchi pattern.

sequence_index: (required) *NX_POSINT* <=

binning: (optional) *NXprocess*

Binning i.e. downsampling of the pattern.

sequence_index: (required) *NX_POSINT* <=

parameter: (optional) *NXprocess*

Specific parameter relevant only for certain algorithms used

sequence_index: (required) *NX_POSINT* <=

EM_EBSD_CRYSTAL_STRUCTURE_MODEL: (required) *NXem_ebsd_crystal_structure_model*

crystallographic_database_identifier: (recommended) *NX_CHAR* <=

crystallographic_database: (recommended) *NX_CHAR* <=

unit_cell_abc: (required) *NX_FLOAT* <=

unit_cell_alphabetagamma: (required) *NX_FLOAT* <=

space_group: (recommended) *NX_CHAR* <=

phase_identifier: (required) *NX_UINT* <=

phase_name: (recommended) *NX_CHAR* <=

atom_identifier: (recommended) *NX_CHAR* <=

atom: (recommended) *NX_UINT* <=

atom_positions: (recommended) *NX_FLOAT* <=

atom_occupancy: (recommended) *NX_FLOAT* <=

number_of_planes: (recommended) *NX_UINT* <=

plane_miller: (recommended) *NX_NUMBER* <=

dspacing: (recommended) *NX_FLOAT* <=

relative_intensity: (recommended) *NX_FLOAT* <=

region_of_interest: (required) *NXprocess*

An overview of the entire area which was scanned. For details about what defines the image contrast inspect descriptor. **descriptor**: (required) *NX_CHAR*

Descriptor representing the image contrast.

Any of these values:

- `normalized_band_contrast`
- `normalized_confidence_index`

roi: (required) *NXdata*

Container holding a default plot of the region on the sample investigated with EBSD. **@signal**: (required) *NX_CHAR <=*

@axes: (required) *NX_CHAR <=*

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR <=*

data: (required) *NX_NUMBER* (Rank: 2, Dimensions: [n_y, n_x]) {units=*NX_UNITLESS*} *<=*

Descriptor values displaying the ROI.

@long_name: (required) *NX_CHAR <=*

Signal

axis_y: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

Calibrated center of mass of the pixel along the slow axis.

@long_name: (required) *NX_CHAR*

Label for the y axis

axis_x: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_LENGTH*}

Calibrated center of mass of the pixel along the fast axis.

@long_name: (required) *NX_CHAR*

Label for the x axis

PROCESS: (optional) *NXprocess*

Default inverse pole figure (IPF) plot of the data specific for each phase. No ipf_mapID instances for non-indexed scan points as these are by definition assigned the null phase with phase_identifier 0.

The IPF mapping is interpolated from the scan point data mapping onto a rectangular domain with square pixels and the orientations colored according to the coloring scheme used in the respective ipf_color_modelID/program.

The main purpose of the ipf_mapID group is not to keep raw data or scan point related data but offer a default way how a research data management system can display a preview of the dataset so that users working with the RDMS can get an overview of the dataset.

This matches the first aim of NXem_ebsd which is foremost to bring colleagues and users of EBSD together to discuss which pieces of information need to be stored together. We are convinced a step-by-step design and community-driven discussion about which pieces of information should and/or need to be included is a practical strategy to work towards an interoperable description and data model for exchanging data from EBSD between different tools and research data management systems (RDMS).

With this design the individual RDMS solutions and tools can still continue to support specific custom data analyses workflow and routes but at least there is then one common notation of understanding whereby also users not necessarily expert in all the details of the EBSD story can understand better these data and thus eventually this can motivate data reuse and repurposing.

It is important to mention that we cannot assume, at least for now, that the parser which writes to an NXem_ebsd-compliant file is also responsible or capable at all of computing the inverse pole figure color keys and maps itself. This cannot be assumed working because this mapping of orientation data uses involved mathematical algorithms and functions which not every tools used in the EBSD community is capable of using or is for sure not using in exactly the same way.

Currently, we assume it is the responsibilty of the tool used which generated the data under on_the_fly_indexing to compute these plots and deliver these to the parser.

Specific case studies have been explored by the experiment team of Area B of the FAIRmat project to realize and implement such mapping.

The first case study uses the H5OINA format and the pyxem/orix library. As orix is a Python library, the coloring is performed by the em_om parser.

The second case study uses MTex and its EBSD color coding model. As MTex is a Matlab tool, an intermediate format is written from MTex first which stores these pieces of information. The parser then pulls these data from the intermediate Matlab-agnostic representation and supplements the file with missing pieces of information as it is required by NXem_ebsd.

The third case study shows how a generic set of Kikuchi pattern can be loaded with the em_om parser. The pattern are loaded directly from a ZIP file and mapped to an simulation image section for now.

The fourth case study uses the DREAM.3D package which provides an own set of EBSD data post-processing procedures. DREAM.3D documents the processing steps with a pipeline file which is stored inside DREAM.3D output files. In this case study, the parser reads the DREAM.3D file and maps data relevant from the perspective of NXem_ebsd plus adds relevant IPF color maps as they were computed by DREAM.3D. Given that in this case the origin of the data is the DREAM.3D file again provenance is kept and more details can be followed upon when resolving origin.

These examples offer a first set of suggestions on how to make EBSD data injectable into research data management system using schemes which themselves are agnostic to the specific RDMS and interoperable. Steps of collecting the raw data and post-processing these with custom scripts like MTex or commercial tools so far are mainly undocumented. The limitation is that a program which consumes results or dump files from these tools may not have necessarily all the sufficient information available to check if the injected orientation data and color models are matching the conventions which a user or automated system has injected into an electronic lab notebook from which currently the em_om parser collects the conventions and stores them into this NXem_ebsd instance. The immediate benefit of the here presented NXem_ebsd concept though is that the conventions and reference frame definitions are expected in an ELN-agnostic representation to make NXem_ebsd a generally useful data scheme for EBSD.

Ideally, the em_om parser would load convention-compliant EBSD data and use subsequently a community library to transcode/convert orientation conventions and parameterized orientation values. Thereafter, convention- compliant default plot(s) could be created that would be truely interoperable.

However, given the variety of post-processing tools available surplus the fact that these are not usually executed along standardized post-processing workflows which perform exactly the same algorithmic steps, this is currently not a practically implementable option. Indeed, first developers who wish to implement this would first have to create a library for performing such tasks, mapping generally between conventions, i.e. map and rotate coordinate systems at the parser level.

The unfortunate situation in EBSD is that due to historical reasons and competitive strategies, different players in the field have implemented (slightly) different approaches each of which misses some part of a complete workflow description which is behind EBSD analyses: Sample preparation, measurement, indexing, post-processing, paper…

The here exemplified default plot do not so far apply relevant rotations but takes the orientation values as they come from the origin and using coloring them as they come. It is thus the scientists responsibility to enter and check if the respective dataset is rotation-conventions-wise consistent and fit for a particular task.

Ideally, with all conventions defined it can be possible to develop a converter which rotates the input data. This application definition does not assume this and users should be aware of this limitation.

The key point is that the conventions however are captured and this is the most important step to the development of such a generic transcoder for creating interoperable EBSD datasets.

Currently the conventions remain in the mind or manual lab book of the respective scientists or technicians instead of getting stored and communicated with research papers that are written based on specific dataset, i.e. database entries.

The default gridded representation of the data should not be misinterpreted as the only possible way how EBSD data and OIM maps can be created!

Indeed, the most general case is that patterns are collected for scan points. The scan generator of an electron microscope is instructed to steer the beam in such a way across the specimen surface that the beam illuminates certain positions for a certain amount time (usually equally-spaced and spending about the same amount of time at each position).

Therefore, scan positions can be due to such regular flight plans and represent sampling on lines, line stacks, rectangular regions-of- interests, but also could instruct spiral, random, or adaptive scans instead of tessellations with square or hexagonal pixels.

The majority of EBSD maps is though is reporting results for a regular grid (square, hexagon). What matters though in terms of damage induced by the electron beam and signal quality is the real electron dose history, i.e. for how long the beam exposed which location of the specimen. Especially when electron charging occurs (i.e. an excess amount of charge accumulates due to e.g. poor conducting away of this charge or an improper mounting, too high dose, etc. such details are relevant.

Specifically, the default visualization is an inverse pole-figure (IPF) map with the usual RGB color coding. Different strategies and normalization schemes are in use to define such color coding.

Finally, we should mention that each ipf_map represents data for scan points indexed as one phase. The alias/name of this phase should be stored in phase_name, the phase_identifier give an ID which must not be zero as this value is reserved for non-indexed / null model scan points.

phase_identifier: (required) *NX_UINT* {units=*NX_UNITLESS*}

Specifying which phase this IPF mapping visualizes.

phase_name: (required) *NX_CHAR*

Alias/name for the phase whose indexed scan points are displayed.

description: (optional) *NX_CHAR*

Which IPF definition computation according to backend.

projection_direction: (required) *NX_NUMBER* (Rank: 1, Dimensions: [3]) {units=*NX_UNITLESS*}

Along which axis to project? Typically [0, 0, 1] is chosen.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*}

Bitdepth used for the RGB color model. Usually 8 bit.

PROGRAM: (required) *NXprogram*

The tool/implementation used for creating the IPF color map from the orientation data. Effectively, this program is the backend which performs the computation of the inverse pole figure mappings which can be for some use cases the parser. Consider the explanations in the docstring of the ipf_mapID group. **program**: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

ipf_rgb_map: (required) *NXdata*

The RGB image which represents the IPF map.

@signal: (required) *NX_CHAR <=*

@axes: (required) *NX_CHAR <=*

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR <=*

data: (required) *NX_UINT* (Rank: 3, Dimensions: [n_y, n_x, 3]) {units=*NX_UNITLESS*}

RGB array, with resolution per fastest changing value defined by bitdepth. **@long_name**: (required) *NX_CHAR*

IPF color-coded orientation mapping

axis_y: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

Calibrated center of mass of the pixel along the slow axis.

@long_name: (required) *NX_CHAR*

Label for the y axis

axis_x: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_x])

Calibrated center of mass of the pixel along the fast axis.

@long_name: (required) *NX_CHAR*

Label for the x axis

ipf_rgb_color_model: (required) *NXdata*

For each stereographic standard triangle (SST), i.e. a rendering of the fundamental zone of the crystal-symmetry-reduced orientation space SO3, it is possible to define a color model which assigns each point in the fundamental zone a color. Different mapping models are in use and implement (slightly) different scaling relations. Differences are which base colors of the RGB color model are placed in which extremal position of the SST and where the white point is located. For further details see:

- [G. Nolze et al.](https://doi.org/10.1107/S1600576716012942)
- Srikanth Patala and coworkers"”" work and of others.

Details are implementation-specific and not standardized yet. Given that the SST has a complicated geometry, it cannot yet be visualized using tools like H5Web, which is why for now the em_om parsers takes a rasterized image which is rendered by the backend tool. **@signal**: (required) *NX_CHAR <=*

@axes: (required) *NX_CHAR <=*

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR <=*

data: (required) *NX_UINT* (Rank: 3, Dimensions: [n_y, n_x, 3]) {units=*NX_UNITLESS*}

RGB array, with resolution per fastest changing value defined by bitdepth.

@long_name: (required) *NX_CHAR*

IPF color key in stereographic standard triangle (SST)

axis_y: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_ANY*}

Pixel coordinate along the slow axis.

@long_name: (required) *NX_CHAR*

Label for the y axis

axis_x: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_ANY*}

Pixel coordinate along the fast axis.

@long_name: (required) *NX_CHAR*

Label for the x axis

correlation: (optional) *NXprocess* <=

This application definition also enables to describe a workflow where several EBSD datasets are not only documented but also correlated based on time, position (spatial), or both (spatiotemporal).

Spatial correlations between repetitively characterized regions-of-interests are typically correlated using image registration and alignment algorithms. For this typically so-called landmarks are used. These can be grains with a very large size or specific shape, i.e. grains which are qualitatively different enough to be used as a guide how images are shifted relative to one another. Other commonly used landmarks are fiducial marks which are milled into the specimen surface using focus-ion beam milling and/or various types of indentation methods.

As far as the same physical region-of-interest is just measured several times, the additional issue of the depth increment is not a concern. However, correct assumptions for the depth increment, amount of material removed along the milling direction is relevant for accurate and precise three-dimensional (serial-sectioning) correlations. For these studies it can be tricky though to assume or estimate useful depth increments. Different strategies have been proposed like calibrations, wedged-shaped landmarks and computer simulation assisted assumption making.

Despite the use of landmarks, there are many practical issues which make the processing of correlations imprecise and inaccurate. Among these are drift and shift of the specimen, instabilities of the holder, the beam, irrespective of the source of the drift, charging effects, here specifically causing local image distortions and rotations which may require special processing algorithms to reduce such imprecisions.

Time correlations face all of the above-mentioned issues surplus the challenge that specific experimental protocols have to be used to ensure the material state is observed at specific physical time. The example of quasi in-situ characterization of crystal growth phenomena, a common topic in engineering or modern catalysis research makes it necessary to consider that e.g. the target value for the desired annealing temperature is not just gauged based on macroscopic arguments but considers that transient effects take place. Heating or quenching a sample might thus might not have been executed under conditions in the interaction volume as they are documented and/or assumed.

These issue cause that correlations have an error margin as to how accurately respective datasets were not only just synced based on the geometry of the region-of-interests and the time markers but also to asssure which physical conditions the specimen experienced over the course of the measurements.

The fourth example of the em_om reference implementation explores the use of the correlation group with a serial-sectioning datasets that was collected by the classical Inconel 100 dataset collected by M. D. Uchic and colleagues (M. Groeber M, Haley BK, Uchic MD, Dimiduk DM, Ghosh S 3d reconstruction and characterization of polycrystalline microstructures using a fib-sem system data set. Mater Charac 2006, 57 259–273. 10.1016/j.matchar.2006.01.019M).

This dataset was specifically relevant in driving forward the implementation of the DREAM.3D software. DREAM.3D is an open-source software project for post-processing and reconstructing, i.e. correlating sets of orientation microscopy data foremost spatially. One focus of the software is the (post-)processing of EBSD datasets. Another cutting edge tool with similar scope but a commercial solution by Bruker is QUBE which was developed by P. Konijnenberg and coworkers.

Conceptually, software like DREAM.3D supports users with creating linear workflows of post-processing tasks. Workflows can be instructed via the graphical user interface or via so-called pipeline processing via command line calls. DREAM.3D is especially useful because its internal system documents all input, output, and parameter of the processing steps. This makes DREAM.3D a good candidate to interface with tools like em_om parser. Specifically, DREAM.3D documents numerical results via a customized HDF5 file format called DREAM3D. Workflow steps and settings are stored as nested dictionaries in JSON syntax inside a supplementary JSON file or alongside the data in the DREAM3D file. DREAM.3D has a few hundred algorithms implemented. These are called filters in DREAM.3D terminology.

Users configure a workflow which instructs DREAM.3D to send the data through a chain of predefined and configured filters. Given that for each analysis the filter is documented via its version tags surplus its parameter and setting via a controlled vocabulary, interpreting the content of a DREAM3D HDF5 file is possible in an automated manner using a parser. This makes DREAM.3D analyses repeatable and self-descriptive. A key limitation though is that most frequently the initial set of input data come from commercial files like ANG. This missing link between the provenance of these input files, their associated creation as electron microscope session, is also what NXem_ebsd solves.

Nevertheless, as this can be solved with e.g. NXem_ebsd we are convinced that the DREAM.3D and the em_om parser can work productively together to realize RDMS-agnostic parsing of serial-section analyses.

The internal documentation of the DREAM.3D workflow also simplifies the provenance tracking represented by an instance of NXem_ebsd as not every intermediate results has to be stored. Therefore, the fourth example focuses on the key result obtained from DREAM.3D - the reconstructed and aligned three-dimensional orientation map.

Usually, this result is the starting point for further post-processing and characterization of structural features. As here orientation microscopy is insofar scale invariant using DREAM.3D, NXem_ebsd, and em_om should be useful for different characterization methods, such as EBSD, Transmission Kikuchi Diffraction (TKD), Automated Crystal Orientation Mapping (ACOM), Nanobeam Electron Diffraction (using commercial systems like NanoMegas ASTAR) or open-source implementations of these techniques (such as via pyxem/orix).

The result of orientation microscopy methods are maps of local orientation and thermodynamic phase (crystal structure) pieces of information. Virtually all post-processing of such results for structural features includes again a workflow of steps which are covered though by the NXms partner application definition. The respective source of the data in an instance of NXms can again be a link or reference to an instance of NXem_ebsd to complete the chain of provenance. **sequence_index**: (required) *NX_POSINT* <=

EM_EBSD_CRYSTAL_STRUCTURE_MODEL: (required) *NXem_ebsd_crystal_structure_model*

crystallographic_database_identifier: (recommended) *NX_CHAR* <=

crystallographic_database: (recommended) *NX_CHAR* <=

unit_cell_abc: (required) *NX_FLOAT* <=

unit_cell_alphabetagamma: (required) *NX_FLOAT* <=

space_group: (recommended) *NX_CHAR* <=

phase_identifier: (required) *NX_UINT* <=

phase_name: (recommended) *NX_CHAR* <=

region_of_interest: (required) *NXprocess*

An overview of the entire reconstructed volume. For details about what defines the image contrast inspect descriptor. **descriptor**: (required) *NX_CHAR*

Descriptor representing the image contrast.

roi: (required) *NXdata*

Container holding a default plot of the reconstructed volume.

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR* <=

data: (required) *NX_NUMBER* (Rank: 3, Dimensions: [n_z, n_y, n_x]) {units=*NX_UNITLESS*} <=

Descriptor values displaying the ROI.

@long_name: (required) *NX_CHAR* <=

Signal

axis_z: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_z]) {units=*NX_LENGTH*}

Calibrated center of mass of the pixel along the slow axis.

@long_name: (required) *NX_CHAR*

Label for the z axis

axis_y: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

Calibrated center of mass of the pixel along the fast axis.

@long_name: (required) *NX_CHAR*

Label for the y axis

axis_x: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_LENGTH*}

Calibrated center of mass of the pixel along the fastest axis.

@long_name: (required) *NX_CHAR*

Label for the x axis

PROCESS: (optional) *NXprocess*

Default inverse pole figure (IPF) plot of the data specific for each phase. No ipf_mapID instances for non-indexed scan points as these are by definition assigned the null phase with phase_identifier 0. The same comments apply as to the two-dimensional representation. **phase_identifier**: (required) *NX_UINT* {units=*NX_UNITLESS*}

Specifying which phase this IPF mapping visualizes.

phase_name: (required) *NX_CHAR*

Alias/name for the phase whose indexed scan points are displayed.

description: (optional) *NX_CHAR*

Which IPF definition computation according to backend.

projection_direction: (required) *NX_NUMBER* (Rank: 1, Dimensions: [3]) {units=*NX_UNITLESS*}

Along which axis to project? Typically [0, 0, 1] is chosen.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*}

Bitdepth used for the RGB color model. Usually 8 bit.

PROGRAM: (required) *NXprogram*

The tool/implementation used for creating the IPF color map from the orientation data. Effectively, this program is the backend which performs the computation of the inverse pole figure mappings which can be for some use cases the parser. Consider the explanations in the docstring of the ipf_mapID group. **program**: (required) *NX_CHAR <=*

@version: (required) *NX_CHAR <=*

ipf_rgb_map: (required) *NXdata*

The RGB image which represents the IPF map.

@signal: (required) *NX_CHAR <=*

@axes: (required) *NX_CHAR <=*

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR <=*

data: (required) *NX_UINT* (Rank: 4, Dimensions: [n_z, n_y, n_x, 3]) {units=*NX_UNITLESS*}

RGB array, with resolution per fastest changing value defined by bitdepth. **@long_name**: (required) *NX_CHAR*

IPF color-coded orientation mapping

axis_z: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_z]) {units=*NX_LENGTH*}

Calibrated center of mass of the pixel along the slow axis.

@long_name: (required) *NX_CHAR*

Label for the z axis

axis_y: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

Calibrated center of mass of the pixel along the faster axis.

@long_name: (required) *NX_CHAR*

Label for the y axis

axis_x: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_x])

Calibrated center of mass of the pixel along the fastest axis.

@long_name: (required) *NX_CHAR*

Label for the x axis

ipf_rgb_color_model: (required) *NXdata*

Same comments as for the two-dimensional case apply.

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_indices: (required) *NX_CHAR*

title: (required) *NX_CHAR* <=

data: (required) *NX_UINT* (Rank: 3, Dimensions: [n_y, n_x, 3]) {units=*NX_UNITLESS*}

RGB array, with resolution per fastest changing value defined by bitdepth.

@long_name: (required) *NX_CHAR*

IPF color key in stereographic standard triangle (SST)

axis_y: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_ANY*}

Pixel coordinate along the slow axis.

@long_name: (required) *NX_CHAR*

Label for the y axis

axis_x: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_ANY*}

Pixel coordinate along the fast axis.

@long_name: (required) *NX_CHAR*

Label for the x axis

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXem_ebsd/ENTRY-group*
- */NXem_ebsd/ENTRY/conventions-group*
- */NXem_ebsd/ENTRY/conventions/detector_reference_frame-group*
- */NXem_ebsd/ENTRY/conventions/detector_reference_frame/origin-field*
- */NXem_ebsd/ENTRY/conventions/detector_reference_frame/reference_frame_type-field*
- */NXem_ebsd/ENTRY/conventions/detector_reference_frame/xaxis_direction-field*
- */NXem_ebsd/ENTRY/conventions/detector_reference_frame/yaxis_direction-field*
- */NXem_ebsd/ENTRY/conventions/detector_reference_frame/zaxis_direction-field*
- */NXem_ebsd/ENTRY/conventions/gnomonic_projection_reference_frame-group*
- */NXem_ebsd/ENTRY/conventions/gnomonic_projection_reference_frame/origin-field*
- */NXem_ebsd/ENTRY/conventions/gnomonic_projection_reference_frame/reference_frame_type-field*
- */NXem_ebsd/ENTRY/conventions/gnomonic_projection_reference_frame/xaxis_direction-field*
- */NXem_ebsd/ENTRY/conventions/gnomonic_projection_reference_frame/yaxis_direction-field*
- */NXem_ebsd/ENTRY/conventions/gnomonic_projection_reference_frame/zaxis_direction-field*
- */NXem_ebsd/ENTRY/conventions/pattern_centre-group*
- */NXem_ebsd/ENTRY/conventions/pattern_centre/xaxis_boundary_convention-field*
- */NXem_ebsd/ENTRY/conventions/pattern_centre/xaxis_normalization_direction-field*
- */NXem_ebsd/ENTRY/conventions/pattern_centre/yaxis_boundary_convention-field*
- */NXem_ebsd/ENTRY/conventions/pattern_centre/yaxis_normalization_direction-field*
- */NXem_ebsd/ENTRY/conventions/processing_reference_frame-group*
- */NXem_ebsd/ENTRY/conventions/processing_reference_frame/origin-field*
- */NXem_ebsd/ENTRY/conventions/processing_reference_frame/reference_frame_type-field*
- */NXem_ebsd/ENTRY/conventions/processing_reference_frame/xaxis_alias-field*
- */NXem_ebsd/ENTRY/conventions/processing_reference_frame/xaxis_direction-field*
- */NXem_ebsd/ENTRY/conventions/processing_reference_frame/yaxis_alias-field*
- */NXem_ebsd/ENTRY/conventions/processing_reference_frame/yaxis_direction-field*
- */NXem_ebsd/ENTRY/conventions/processing_reference_frame/zaxis_alias-field*
- */NXem_ebsd/ENTRY/conventions/processing_reference_frame/zaxis_direction-field*
- */NXem_ebsd/ENTRY/conventions/rotation_conventions-group*
- */NXem_ebsd/ENTRY/conventions/rotation_conventions/axis_angle_convention-field*
- */NXem_ebsd/ENTRY/conventions/rotation_conventions/euler_angle_convention-field*
- */NXem_ebsd/ENTRY/conventions/rotation_conventions/orientation_parameterization_sign_convention-field*
- */NXem_ebsd/ENTRY/conventions/rotation_conventions/rotation_convention-field*
- */NXem_ebsd/ENTRY/conventions/rotation_conventions/three_dimensional_rotation_handedness-field*

- */NXem_ebsd/ENTRY/conventions/sample_reference_frame-group*
- */NXem_ebsd/ENTRY/conventions/sample_reference_frame/origin-field*
- */NXem_ebsd/ENTRY/conventions/sample_reference_frame/reference_frame_type-field*
- */NXem_ebsd/ENTRY/conventions/sample_reference_frame/xaxis_direction-field*
- */NXem_ebsd/ENTRY/conventions/sample_reference_frame/yaxis_direction-field*
- */NXem_ebsd/ENTRY/conventions/sample_reference_frame/zaxis_direction-field*
- */NXem_ebsd/ENTRY/correlation-group*
- */NXem_ebsd/ENTRY/correlation/EM_EBSD_CRYSTAL_STRUCTURE_MODEL-group*
- */NXem_ebsd/ENTRY/correlation/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/crystallographic_database-field*
- */NXem_ebsd/ENTRY/correlation/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/crystallographic_database_identifier-field*
- */NXem_ebsd/ENTRY/correlation/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/phase_identifier-field*
- */NXem_ebsd/ENTRY/correlation/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/phase_name-field*
- */NXem_ebsd/ENTRY/correlation/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/space_group-field*
- */NXem_ebsd/ENTRY/correlation/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/unit_cell_abc-field*
- */NXem_ebsd/ENTRY/correlation/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/unit_cell_alphabetagamma-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS-group*
- */NXem_ebsd/ENTRY/correlation/PROCESS/bitdepth-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS/description-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_color_model-group*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_color_model/axis_x-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_color_model/axis_x@long_name-attribute*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_color_model/axis_y-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_color_model/axis_y@long_name-attribute*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_color_model/data-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_color_model/data@long_name-attribute*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_color_model/title-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_color_model@axes-attribute*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_color_model@AXISNAME_indices-attribute*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_color_model@signal-attribute*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_map-group*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_map/axis_x-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_map/axis_x@long_name-attribute*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_map/axis_y-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_map/axis_y@long_name-attribute*

- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_map/axis_z-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_map/axis_z@long_name-attribute*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_map/data-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_map/data@long_name-attribute*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_map/title-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_map@axes-attribute*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_map@AXISNAME_indices-attribute*
- */NXem_ebsd/ENTRY/correlation/PROCESS/ipf_rgb_map@signal-attribute*
- */NXem_ebsd/ENTRY/correlation/PROCESS/phase_identifier-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS/phase_name-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS/PROGRAM-group*
- */NXem_ebsd/ENTRY/correlation/PROCESS/PROGRAM/program-field*
- */NXem_ebsd/ENTRY/correlation/PROCESS/PROGRAM/program@version-attribute*
- */NXem_ebsd/ENTRY/correlation/PROCESS/projection_direction-field*
- */NXem_ebsd/ENTRY/correlation/region_of_interest-group*
- */NXem_ebsd/ENTRY/correlation/region_of_interest/descriptor-field*
- */NXem_ebsd/ENTRY/correlation/region_of_interest/roi-group*
- */NXem_ebsd/ENTRY/correlation/region_of_interest/roi/axis_x-field*
- */NXem_ebsd/ENTRY/correlation/region_of_interest/roi/axis_x@long_name-attribute*
- */NXem_ebsd/ENTRY/correlation/region_of_interest/roi/axis_y-field*
- */NXem_ebsd/ENTRY/correlation/region_of_interest/roi/axis_y@long_name-attribute*
- */NXem_ebsd/ENTRY/correlation/region_of_interest/roi/axis_z-field*
- */NXem_ebsd/ENTRY/correlation/region_of_interest/roi/axis_z@long_name-attribute*
- */NXem_ebsd/ENTRY/correlation/region_of_interest/roi/data-field*
- */NXem_ebsd/ENTRY/correlation/region_of_interest/roi/data@long_name-attribute*
- */NXem_ebsd/ENTRY/correlation/region_of_interest/roi/title-field*
- */NXem_ebsd/ENTRY/correlation/region_of_interest/roi@axes-attribute*
- */NXem_ebsd/ENTRY/correlation/region_of_interest/roi@AXISNAME_indices-attribute*
- */NXem_ebsd/ENTRY/correlation/region_of_interest/roi@signal-attribute*
- */NXem_ebsd/ENTRY/correlation/sequence_index-field*
- */NXem_ebsd/ENTRY/definition-field*
- */NXem_ebsd/ENTRY/end_time-field*
- */NXem_ebsd/ENTRY/experiment-group*
- */NXem_ebsd/ENTRY/experiment/acquisition-group*
- */NXem_ebsd/ENTRY/experiment/acquisition/origin-field*
- */NXem_ebsd/ENTRY/experiment/acquisition/origin@version-attribute*

- */NXem_ebsd/ENTRY/experiment/acquisition/path-field*
- */NXem_ebsd/ENTRY/experiment/acquisition/sequence_index-field*
- */NXem_ebsd/ENTRY/experiment/calibration-group*
- */NXem_ebsd/ENTRY/experiment/calibration/origin-field*
- */NXem_ebsd/ENTRY/experiment/calibration/origin@version-attribute*
- */NXem_ebsd/ENTRY/experiment/calibration/path-field*
- */NXem_ebsd/ENTRY/experiment/calibration/sequence_index-field*
- */NXem_ebsd/ENTRY/experiment/indexing-group*
- */NXem_ebsd/ENTRY/experiment/indexing/background_correction-group*
- */NXem_ebsd/ENTRY/experiment/indexing/background_correction/sequence_index-field*
- */NXem_ebsd/ENTRY/experiment/indexing/binning-group*
- */NXem_ebsd/ENTRY/experiment/indexing/binning/sequence_index-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL-group*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/atom-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/atom_identifier-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/atom_occupancy-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/atom_positions-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/crystallographic_database-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/crystallographic_database_identifier-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/dspacing-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/number_of_planes-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/phase_identifier-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/phase_name-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/plane_miller-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/relative_intensity-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/space_group-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/unit_cell_abc-field*
- */NXem_ebsd/ENTRY/experiment/indexing/EM_EBSD_CRYSTAL_STRUCTURE_MODEL/unit_cell_alphabetagamma-field*
- */NXem_ebsd/ENTRY/experiment/indexing/hit_rate-field*
- */NXem_ebsd/ENTRY/experiment/indexing/method-field*
- */NXem_ebsd/ENTRY/experiment/indexing/n_phases_per_scan_point-field*

- /NXem_ebsd/ENTRY/experiment/indexing/on_the_fly_indexing-group
- /NXem_ebsd/ENTRY/experiment/indexing/on_the_fly_indexing/origin-field
- /NXem_ebsd/ENTRY/experiment/indexing/on_the_fly_indexing/origin@version-attribute
- /NXem_ebsd/ENTRY/experiment/indexing/on_the_fly_indexing/path-field
- /NXem_ebsd/ENTRY/experiment/indexing/on_the_fly_indexing/PROGRAM-group
- /NXem_ebsd/ENTRY/experiment/indexing/on_the_fly_indexing/PROGRAM/program-field
- /NXem_ebsd/ENTRY/experiment/indexing/on_the_fly_indexing/PROGRAM/program@version-attribute
- /NXem_ebsd/ENTRY/experiment/indexing/orientation-field
- /NXem_ebsd/ENTRY/experiment/indexing/orientation_parameterization-field
- /NXem_ebsd/ENTRY/experiment/indexing/parameter-group
- /NXem_ebsd/ENTRY/experiment/indexing/parameter/sequence_index-field
- /NXem_ebsd/ENTRY/experiment/indexing/phase_identifier-field
- /NXem_ebsd/ENTRY/experiment/indexing/phase_matching-field
- /NXem_ebsd/ENTRY/experiment/indexing/phase_matching_descriptor-field
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS-group
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/bitdepth-field
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/description-field
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_color_model-group
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_color_model/axis_x-field
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_color_model/axis_x@long_name-attribute
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_color_model/axis_y-field
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_color_model/axis_y@long_name-attribute
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_color_model/data-field
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_color_model/data@long_name-attribute
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_color_model/title-field
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_color_model@axes-attribute
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_color_model@AXISNAME_indices-attribute
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_color_model@signal-attribute
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_map-group
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_map/axis_x-field
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_map/axis_x@long_name-attribute
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_map/axis_y-field
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_map/axis_y@long_name-attribute
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_map/data-field
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_map/data@long_name-attribute
- /NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_map/title-field

- */NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_map@axes-attribute*
- */NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_map@AXISNAME_indices-attribute*
- */NXem_ebsd/ENTRY/experiment/indexing/PROCESS/ipf_rgb_map@signal-attribute*
- */NXem_ebsd/ENTRY/experiment/indexing/PROCESS/phase_identifier-field*
- */NXem_ebsd/ENTRY/experiment/indexing/PROCESS/phase_name-field*
- */NXem_ebsd/ENTRY/experiment/indexing/PROCESS/PROGRAM-group*
- */NXem_ebsd/ENTRY/experiment/indexing/PROCESS/PROGRAM/program-field*
- */NXem_ebsd/ENTRY/experiment/indexing/PROCESS/PROGRAM/program@version-attribute*
- */NXem_ebsd/ENTRY/experiment/indexing/PROCESS/projection_direction-field*
- */NXem_ebsd/ENTRY/experiment/indexing/region_of_interest-group*
- */NXem_ebsd/ENTRY/experiment/indexing/region_of_interest/descriptor-field*
- */NXem_ebsd/ENTRY/experiment/indexing/region_of_interest/roi-group*
- */NXem_ebsd/ENTRY/experiment/indexing/region_of_interest/roi/axis_x-field*
- */NXem_ebsd/ENTRY/experiment/indexing/region_of_interest/roi/axis_x@long_name-attribute*
- */NXem_ebsd/ENTRY/experiment/indexing/region_of_interest/roi/axis_y-field*
- */NXem_ebsd/ENTRY/experiment/indexing/region_of_interest/roi/axis_y@long_name-attribute*
- */NXem_ebsd/ENTRY/experiment/indexing/region_of_interest/roi/data-field*
- */NXem_ebsd/ENTRY/experiment/indexing/region_of_interest/roi/data@long_name-attribute*
- */NXem_ebsd/ENTRY/experiment/indexing/region_of_interest/roi/title-field*
- */NXem_ebsd/ENTRY/experiment/indexing/region_of_interest/roi@axes-attribute*
- */NXem_ebsd/ENTRY/experiment/indexing/region_of_interest/roi@AXISNAME_indices-attribute*
- */NXem_ebsd/ENTRY/experiment/indexing/region_of_interest/roi@signal-attribute*
- */NXem_ebsd/ENTRY/experiment/indexing/scan_point_positions-field*
- */NXem_ebsd/ENTRY/experiment/indexing/sequence_index-field*
- */NXem_ebsd/ENTRY/experiment/indexing/status-field*
- */NXem_ebsd/ENTRY/experiment/time-field*
- */NXem_ebsd/ENTRY/experiment/TRANSFORMATIONS-group*
- */NXem_ebsd/ENTRY/PROGRAM-group*
- */NXem_ebsd/ENTRY/PROGRAM/program-field*
- */NXem_ebsd/ENTRY/PROGRAM/program@version-attribute*
- */NXem_ebsd/ENTRY/simulation-group*
- */NXem_ebsd/ENTRY/simulation/CG_GEODESIC_MESH-group*
- */NXem_ebsd/ENTRY/simulation/IMAGE_SET_EM_KIKUCHI-group*
- */NXem_ebsd/ENTRY/simulation/IMAGE_SET_EM_KIKUCHI/stack-group*
- */NXem_ebsd/ENTRY/simulation/IMAGE_SET_EM_KIKUCHI/stack/axis_x-field*
- */NXem_ebsd/ENTRY/simulation/IMAGE_SET_EM_KIKUCHI/stack/axis_x@long_name-attribute*

- */NXem_ebsd/ENTRY/simulation/IMAGE_SET_EM_KIKUCHI/stack/axis_y-field*
- */NXem_ebsd/ENTRY/simulation/IMAGE_SET_EM_KIKUCHI/stack/axis_y@long_name-attribute*
- */NXem_ebsd/ENTRY/simulation/IMAGE_SET_EM_KIKUCHI/stack/data_counts-field*
- */NXem_ebsd/ENTRY/simulation/IMAGE_SET_EM_KIKUCHI/stack/data_counts@long_name-attribute*
- */NXem_ebsd/ENTRY/simulation/IMAGE_SET_EM_KIKUCHI/stack/pattern_identifier-field*
- */NXem_ebsd/ENTRY/simulation/IMAGE_SET_EM_KIKUCHI/stack/pattern_identifier@long_name-attribute*
- */NXem_ebsd/ENTRY/simulation/IMAGE_SET_EM_KIKUCHI/stack/title-field*
- */NXem_ebsd/ENTRY/simulation/IMAGE_SET_EM_KIKUCHI/stack@axes-attribute*
- */NXem_ebsd/ENTRY/simulation/IMAGE_SET_EM_KIKUCHI/stack@AXISNAME_indices-attribute*
- */NXem_ebsd/ENTRY/simulation/IMAGE_SET_EM_KIKUCHI/stack@signal-attribute*
- */NXem_ebsd/ENTRY/simulation/PROGRAM-group*
- */NXem_ebsd/ENTRY/simulation/PROGRAM/program-field*
- */NXem_ebsd/ENTRY/simulation/PROGRAM/program@version-attribute*
- */NXem_ebsd/ENTRY/simulation/sequence_index-field*
- */NXem_ebsd/ENTRY/start_time-field*
- */NXem_ebsd/ENTRY/USER-group*
- */NXem_ebsd/ENTRY/USER/address-field*
- */NXem_ebsd/ENTRY/USER/affiliation-field*
- */NXem_ebsd/ENTRY/USER/email-field*
- */NXem_ebsd/ENTRY/USER/name-field*
- */NXem_ebsd/ENTRY/USER/orcid-field*
- */NXem_ebsd/ENTRY/USER/orcid_platform-field*
- */NXem_ebsd/ENTRY/USER/role-field*
- */NXem_ebsd/ENTRY/USER/social_media_name-field*
- */NXem_ebsd/ENTRY/USER/social_media_platform-field*
- */NXem_ebsd/ENTRY/USER/telephone_number-field*
- */NXem_ebsd/ENTRY/workflow_description-field*
- */NXem_ebsd/ENTRY/workflow_identifier-field*
- */NXem_ebsd/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXem_ebsd.nxdl.xml

NXem_ebsd_conventions

Status:

base class, extends *NXobject*

Description:

Conventions for rotations and coordinate systems to interpret EBSD data.

This is the main issue which currently is not in all cases documented and thus limits the interoperability and value of collected EBSD data. Not communicating EBSD data with such contextual pieces of information and the use of file formats which do not store this information is the key unsolved problem.

Symbols:

No symbol table

Groups cited:
NXprocess

Structure:

rotation_conventions: (optional) *NXprocess*

Mathematical conventions and materials-science-specific conventions required for interpreting every collection of orientation data. **three_dimensional_rotation_handedness**: (optional) *NX_CHAR*

Convention how a positive rotation angle is defined when viewing from the end of the rotation unit vector towards its origin, i.e. in accordance with convention 2 of DOI: 10.1088/0965-0393/23/8/083501. Counter_clockwise is equivalent to a right-handed choice. Clockwise is equivalent to a left-handed choice.

Any of these values: `undefined` | `counter_clockwise` | `clockwise`

rotation_convention: (optional) *NX_CHAR*

How are rotations interpreted into an orientation according to convention 3 of DOI: 10.1088/0965-0393/23/8/083501.

Any of these values: `undefined` | `passive` | `active`

euler_angle_convention: (optional) *NX_CHAR*

How are Euler angles interpreted given that there are several choices (e.g. ZXZ, XYZ, etc.) according to convention 4 of DOI: 10.1088/0965-0393/23/8/083501. The most frequently used convention is ZXZ which is based on the work of H.-J. Bunge but other conventions are possible.

Any of these values: `undefined` | `zxz`

axis_angle_convention: (optional) *NX_CHAR*

To which angular range is the rotation angle argument of an axis-angle pair parameterization constrained according to convention 5 of DOI: 10.1088/0965-0393/23/8/083501.

Any of these values: `undefined` | `rotation_angle_on_interval_zero_to_pi`

orientation_parameterization_sign_convention: (optional) *NX_CHAR*

Which sign convention is followed when converting orientations between different parameterizations/representations according to convention 6 of DOI: 10.1088/0965-0393/23/8/083501.

Any of these values: `undefined` | `p_plus_one` | `p_minus_one`

processing_reference_frame: (optional) *NXprocess*

Details about eventually relevant named directions that may give reasons for anisotropies. The classical example is cold-rolling where one has to specify which directions (rolling, transverse, and normal) align how with the direction of the base vectors of the sample_reference_frame.

reference_frame_type: (optional) *NX_CHAR*

Type of coordinate system and reference frame according to convention 1 of DOI: 10.1088/0965-0393/23/8/083501.

Any of these values:

- `undefined`
- `right_handed_cartesian`
- `left_handed_cartesian`

xaxis_direction: (optional) *NX_CHAR*

Direction of the positively pointing x-axis base vector of the processing_reference_frame. We assume the configuration is inspected by looking towards the sample surface from a position that is located behind the detector.

Any of these values:

- `undefined`
- `north`
- `east`
- `south`
- `west`
- `in`
- `out`

xaxis_alias: (optional) *NX_CHAR*

Name or alias assigned to the x-axis base vector, e.g. rolling direction.

yaxis_direction: (optional) *NX_CHAR*

Direction of the positively pointing y-axis base vector of the processing_reference_frame. We assume the configuration is inspected by looking towards the sample surface from a position that is located behind the detector. For further information consult also the help info for the xaxis_direction field.

Any of these values:

- `undefined`
- `north`
- `east`
- `south`
- `west`
- `in`
- `out`

yaxis_alias: (optional) *NX_CHAR*

Name or alias assigned to the y-axis base vector, e.g. transverse direction.

zaxis_direction: (optional) *NX_CHAR*

Direction of the positively pointing z-axis base vector of the processing_reference frame. We assume the configuration is inspected by looking towards the sample surface from a position that is located behind the detector. For further information consult also the help info for the xaxis_direction field.

Any of these values:

- `undefined`
- `north`
- `east`
- `south`
- `west`
- `in`
- `out`

zaxis_alias: (optional) *NX_CHAR*

Name or alias assigned to the z-axis base vector, e.g. normal direction.

origin: (optional) *NX_CHAR*

Location of the origin of the processing_reference_frame. This specifies the location Xp = 0, Yp = 0, Zp = 0. Assume regions-of-interest in this reference frame form a rectangle or cuboid. Edges are interpreted by inspecting the direction of their outer unit normals (which point either parallel or antiparallel) along respective base vector direction of the reference frame.

Any of these values:

- `undefined`
- `front_top_left`
- `front_top_right`
- `front_bottom_right`
- `front_bottom_left`
- `back_top_left`
- `back_top_right`
- `back_bottom_right`
- `back_bottom_left`

sample_reference_frame: (optional) *NXprocess*

Details about the sample/specimen reference frame.

reference_frame_type: (optional) *NX_CHAR*

Type of coordinate system and reference frame according to convention 1 of DOI: 10.1088/0965-0393/23/8/083501. The reference frame for the sample surface reference is used for identifying positions on a (virtual) image which is formed by information collected from an electron beam scanning the sample surface. We assume the configuration is inspected by looking towards the sample surface from a position that is located behind the detector. Reference DOI: 10.1016/j.matchar.2016.04.008 The sample surface reference frame has coordinates Xs, Ys, Zs. In three dimensions these coordinates are not necessarily located on the surface of the sample as there are multiple faces/sides of the sample. Most frequently though the coordinate system here is used to define the surface which the electron beam scans.

Any of these values:

- `undefined`
- `right_handed_cartesian`
- `left_handed_cartesian`

xaxis_direction: (optional) *NX_CHAR*

Direction of the positively pointing x-axis base vector of the sample surface reference frame. We assume the configuration is inspected by looking towards the sample surface from a position that is located behind the detector. Different tools assume that different strategies can be used and are perceived as differently convenient to enter details about coordinate system definitions. In this ELN users have to possibility to fill in what they assume is sufficient to define the coordinate system directions unambiguously. Software which works with this user input needs to offer parsing capabilities which detect conflicting input and warn accordingly.

Any of these values:

- `undefined`
- `north`
- `east`
- `south`
- `west`
- `in`
- `out`

yaxis_direction: (optional) *NX_CHAR*

Direction of the positively pointing y-axis base vector of the sample surface reference frame. We assume the configuration is inspected by looking towards the sample surface from a position that is located behind the detector. For further information consult also the help info for the xaxis_direction field.

Any of these values:

- `undefined`
- `north`
- `east`
- `south`
- `west`

- in
- out

zaxis_direction: (optional) *NX_CHAR*

Direction of the positively pointing z-axis base vector of the sample surface reference frame. We assume the configuration is inspected by looking towards the sample surface from a position that is located behind the detector. For further information consult also the help info for the xaxis_direction field.

Any of these values:

- undefined
- north
- east
- south
- west
- in
- out

origin: (optional) *NX_CHAR*

Location of the origin of the sample surface reference frame. This specifies the location Xs = 0, Ys = 0, Zs = 0. Assume regions-of-interest in this reference frame form a rectangle or cuboid. Edges are interpreted by inspecting the direction of their outer unit normals (which point either parallel or antiparallel) along respective base vector direction of the reference frame.

Any of these values:

- undefined
- front_top_left
- front_top_right
- front_bottom_right
- front_bottom_left
- back_top_left
- back_top_right
- back_bottom_right
- back_bottom_left

detector_reference_frame: (optional) *NXprocess*

Details about the detector reference frame.

reference_frame_type: (optional) *NX_CHAR*

Type of coordinate system/reference frame used for identifying positions in detector space Xd, Yd, Zd, according to DOI: 10.1016/j.matchar.2016.04.008.

Any of these values:

- undefined
- right_handed_cartesian

- `left_handed_cartesian`

xaxis_direction: (optional) *NX_CHAR*

Direction of the positively pointing x-axis base vector of the detector space reference frame. We assume the configuration is inspected by looking towards the sample surface from a position that is located behind the detector. Different tools assume that different strategies can be used and are perceived as differently convenient to enter details about coordinate system definitions. In this ELN users have to possibility to fill in what they assume is sufficient to define the coordinate system directions unambiguously. Software which works with this user input needs to offer parsing capabilities which detect conflicting input and warn accordingly.

Any of these values:

- `undefined`
- `north`
- `east`
- `south`
- `west`
- `in`
- `out`

yaxis_direction: (optional) *NX_CHAR*

Direction of the positively pointing y-axis base vector of the detector space reference frame. We assume the configuration is inspected by looking towards the sample surface from a position that is located behind the detector. For further information consult also the help info for the xaxis_direction field.

Any of these values:

- `undefined`
- `north`
- `east`
- `south`
- `west`
- `in`
- `out`

zaxis_direction: (optional) *NX_CHAR*

Direction of the positively pointing z-axis base vector of the detector space reference frame. We assume the configuration is inspected by looking towards the sample surface from a position that is located behind the detector. For further information consult also the help info for the xaxis_direction field.

Any of these values:

- `undefined`
- `north`
- `east`

- `south`
- `west`
- `in`
- `out`

origin: (optional) *NX_CHAR*

Where is the origin of the detector space reference frame located. This is the location of Xd = 0, Yd = 0, Zd = 0. Assume regions-of-interest in this reference frame form a rectangle or cuboid. Edges are interpreted by inspecting the direction of their outer unit normals (which point either parallel or antiparallel) along respective base vector direction of the reference frame.

Any of these values:

- `undefined`
- `front_top_left`
- `front_top_right`
- `front_bottom_right`
- `front_bottom_left`
- `back_top_left`
- `back_top_right`
- `back_bottom_right`
- `back_bottom_left`

gnomonic_projection_reference_frame: (optional) *NXprocess*

Details about the gnomonic projection reference frame.

reference_frame_type: (optional) *NX_CHAR*

Type of coordinate system/reference frame used for identifying positions in the gnomonic projection space Xg, Yg, Zg according to DOI: 10.1016/j.matchar.2016.04.008.

Any of these values:

- `undefined`
- `right_handed_cartesian`
- `left_handed_cartesian`

xaxis_direction: (optional) *NX_CHAR*

Direction of the positively pointing "gnomomic" x-axis base vector when viewing how the diffraction pattern looks on the detector screen. We assume the configuration is inspected by looking towards the sample surface from a position that is located behind the detector. Different tools assume that different strategies can be used and are perceived as differently convenient to enter details about coordinate system definitions. In this ELN users have to possibility to fill in what they assume is sufficient to define the coordinate system directions unambiguously. Software which works with this user input needs to offer parsing capabilities which detect conflicting input and warn accordingly.

Any of these values:

- `undefined`
- `north`
- `east`
- `south`
- `west`
- `in`
- `out`

yaxis_direction: (optional) *NX_CHAR*

Direction of the positively pointing "gnomomic" y-axis base vector when viewing how the diffraction pattern looks on the detector screen. We assume the configuration is inspected by looking towards the sample surface from a position that is located behind the detector. For further information consult also the help info for the xaxis_direction field.

Any of these values:

- `undefined`
- `north`
- `east`
- `south`
- `west`
- `in`
- `out`

zaxis_direction: (optional) *NX_CHAR*

Direction of the positively pointing "gnomomic" z-axis base vector when viewing how the diffraction pattern looks on the detector screen. We assume the configuration is inspected by looking towards the sample surface from a position that is located behind the detector. For further information consult also the help info for the xaxis_direction field.

Any of these values:

- `undefined`
- `north`
- `east`
- `south`
- `west`
- `in`
- `out`

origin: (optional) *NX_CHAR*

Is the origin of the gnomonic coordinate system located where we assume the location of the pattern centre. This is the location Xg = 0, Yg = 0, Zg = 0 according to reference DOI: 10.1016/j.matchar.2016.04.008.

Any of these values: `undefined` | `in_the_pattern_centre`

pattern_centre: (optional) *NXprocess*

Details about the definition of the pattern centre as a special point in the gnomonic projection reference frame. **xaxis_boundary_convention**: (optional) *NX_CHAR*

From which border of the EBSP (in the detector reference frame) is the pattern centre's x-position (PCx) measured? Keywords assume the region-of-interest is defined by a rectangle. We observe this rectangle and inspect the direction of the outer-unit normals to the edges of this rectangle.

Any of these values: `undefined` | `top` | `right` | `bottom` | `left`

xaxis_normalization_direction: (optional) *NX_CHAR*

In which direction are positive values for PCx measured from the specified boundary. Keep in mind that the gnomonic space is in virtually all cases embedded in the detector space. Specifically, the XgYg plane is defined such that it is embedded/laying inside the XdYd plane (of the detector reference frame). When the normalization direction is the same as e.g. the detector x-axis direction, we state that we effectively normalize in fractions of the width of the detector.

The issue with terms like width and height is that these degenerate if the detector region-of-interest is square-shaped. This is why we should better avoid talking about width and height but state how we would measure distances practically with a ruler and how we then measure positive distances.

Any of these values: `undefined` | `north` | `east` | `south` | `west`

yaxis_boundary_convention: (optional) *NX_CHAR*

From which border of the EBSP (in the detector reference frame) is the pattern centre's y-position (PCy) measured? For further details inspect the help button of xaxis_boundary_convention.

Any of these values: `undefined` | `top` | `right` | `bottom` | `left`

yaxis_normalization_direction: (optional) *NX_CHAR*

In which direction are positive values for PCy measured from the specified boundary. For further details inspect the help button of xaxis_normalization_direction.

Any of these values: `undefined` | `north` | `east` | `south` | `west`

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXem_ebsd_conventions/detector_reference_frame-group*
- */NXem_ebsd_conventions/detector_reference_frame/origin-field*
- */NXem_ebsd_conventions/detector_reference_frame/reference_frame_type-field*
- */NXem_ebsd_conventions/detector_reference_frame/xaxis_direction-field*
- */NXem_ebsd_conventions/detector_reference_frame/yaxis_direction-field*
- */NXem_ebsd_conventions/detector_reference_frame/zaxis_direction-field*
- */NXem_ebsd_conventions/gnomonic_projection_reference_frame-group*
- */NXem_ebsd_conventions/gnomonic_projection_reference_frame/origin-field*

- */NXem_ebsd_conventions/gnomonic_projection_reference_frame/reference_frame_type-field*
- */NXem_ebsd_conventions/gnomonic_projection_reference_frame/xaxis_direction-field*
- */NXem_ebsd_conventions/gnomonic_projection_reference_frame/yaxis_direction-field*
- */NXem_ebsd_conventions/gnomonic_projection_reference_frame/zaxis_direction-field*
- */NXem_ebsd_conventions/pattern_centre-group*
- */NXem_ebsd_conventions/pattern_centre/xaxis_boundary_convention-field*
- */NXem_ebsd_conventions/pattern_centre/xaxis_normalization_direction-field*
- */NXem_ebsd_conventions/pattern_centre/yaxis_boundary_convention-field*
- */NXem_ebsd_conventions/pattern_centre/yaxis_normalization_direction-field*
- */NXem_ebsd_conventions/processing_reference_frame-group*
- */NXem_ebsd_conventions/processing_reference_frame/origin-field*
- */NXem_ebsd_conventions/processing_reference_frame/reference_frame_type-field*
- */NXem_ebsd_conventions/processing_reference_frame/xaxis_alias-field*
- */NXem_ebsd_conventions/processing_reference_frame/xaxis_direction-field*
- */NXem_ebsd_conventions/processing_reference_frame/yaxis_alias-field*
- */NXem_ebsd_conventions/processing_reference_frame/yaxis_direction-field*
- */NXem_ebsd_conventions/processing_reference_frame/zaxis_alias-field*
- */NXem_ebsd_conventions/processing_reference_frame/zaxis_direction-field*
- */NXem_ebsd_conventions/rotation_conventions-group*
- */NXem_ebsd_conventions/rotation_conventions/axis_angle_convention-field*
- */NXem_ebsd_conventions/rotation_conventions/euler_angle_convention-field*
- */NXem_ebsd_conventions/rotation_conventions/orientation_parameterization_sign_convention-field*
- */NXem_ebsd_conventions/rotation_conventions/rotation_convention-field*
- */NXem_ebsd_conventions/rotation_conventions/three_dimensional_rotation_handedness-field*
- */NXem_ebsd_conventions/sample_reference_frame-group*
- */NXem_ebsd_conventions/sample_reference_frame/origin-field*
- */NXem_ebsd_conventions/sample_reference_frame/reference_frame_type-field*
- */NXem_ebsd_conventions/sample_reference_frame/xaxis_direction-field*
- */NXem_ebsd_conventions/sample_reference_frame/yaxis_direction-field*
- */NXem_ebsd_conventions/sample_reference_frame/zaxis_direction-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXem_ebsd_conventions.nxdl.xml

NXem_ebsd_crystal_structure_model

Status:

base class, extends *NXobject*

Description:

Crystal structure phase models used for indexing Kikuchi pattern.

This base class contains key metadata relevant to every physical kinematic or dynamic diffraction model to be used for simulating Kikuchi diffraction pattern. The actual indexing of Kikuchi pattern however maybe use different algorithms which build on these simulation results but evaluate different workflows of comparing simulated and measured Kikuchi pattern to make suggestions which orientation is the most likely (if any) for each scan point investigated.

Traditionally Hough transform based indexing has been the most frequently used algorithm. More and more dictionary based alternatives are used. Either way both algorithm need a crystal structure model.

Symbols:

n_hkl: Number of reflectors (Miller crystallographic plane triplets).

n_pos: Number of atom positions.

Groups cited:
none

Structure:

crystallographic_database_identifier: (optional) *NX_CHAR*

Identifier of an entry from crystallographic_database which was used for creating this structure model.

crystallographic_database: (optional) *NX_CHAR*

Name of the crystallographic database to resolve crystallographic_database_identifier e.g. COD or others.

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

space_group: (optional) *NX_CHAR*

Crystallographic space group

is_centrosymmetric: (optional) *NX_BOOLEAN*

True if space group is considered a centrosymmetric one. False if space group is considered a non-centrosymmetric one. Centrosymmetric has all types and combinations of symmetry elements (translation, rotational axis, mirror planes, center of inversion) Non-centrosymmetric compared to centrosymmetric is constrained (no inversion). Chiral compared to non-centrosymmetric is constrained (no mirror planes).

is_chiral: (optional) *NX_BOOLEAN*

True if space group is considered a chiral one. False if space group is consider a non-chiral one.

laue_group: (optional) *NX_CHAR*

Laue group

point_group: (optional) *NX_CHAR*

Point group using International Notation.

unit_cell_class: (optional) *NX_CHAR*

Crystal system

Any of these values:

- `triclinic`
- `monoclinic`
- `orthorhombic`
- `tetragonal`
- `rhombohedral`
- `hexagonal`
- `cubic`

phase_identifier: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Numeric identifier for each phase. The value 0 is reserved for the unknown phase essentially representing the null-model that no phase model was sufficiently significantly confirmed. Consequently, the value 0 must not be used as a phase_identifier.

phase_name: (optional) *NX_CHAR*

Name of the phase/alias.

atom_identifier: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_pos])

Labels for each atom position

atom: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_pos]) {units=*NX_UNITLESS*}

The hash value H is $H = Z + N * 256$ 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)

atom_positions: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [n_pos, 3]) {units=*NX_LENGTH*}

Atom positions x, y, z.

atom_occupancy: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_pos]) {units=*NX_DIMENSIONLESS*}

Relative occupancy of the atom position.

number_of_planes: (optional) *NX_UINT* {units=*NX_UNITLESS*}

How many reflectors are distinguished. Value has to be n_hkl.

plane_miller: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_hkl, 3]) {units=*NX_UNITLESS*}

Miller indices (hkl).

dspacing: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_hkl]) {units=*NX_LENGTH*}

D-spacing.

relative_intensity: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_hkl]) {units=*NX_DIMENSIONLESS*}

Relative intensity of the signal for the plane.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXem_ebsd_crystal_structure_model/atom-field*
- */NXem_ebsd_crystal_structure_model/atom_identifier-field*
- */NXem_ebsd_crystal_structure_model/atom_occupancy-field*
- */NXem_ebsd_crystal_structure_model/atom_positions-field*
- */NXem_ebsd_crystal_structure_model/crystallographic_database-field*
- */NXem_ebsd_crystal_structure_model/crystallographic_database_identifier-field*
- */NXem_ebsd_crystal_structure_model/dspacing-field*
- */NXem_ebsd_crystal_structure_model/is_centrosymmetric-field*
- */NXem_ebsd_crystal_structure_model/is_chiral-field*
- */NXem_ebsd_crystal_structure_model/laue_group-field*
- */NXem_ebsd_crystal_structure_model/number_of_planes-field*
- */NXem_ebsd_crystal_structure_model/phase_identifier-field*
- */NXem_ebsd_crystal_structure_model/phase_name-field*
- */NXem_ebsd_crystal_structure_model/plane_miller-field*
- */NXem_ebsd_crystal_structure_model/point_group-field*
- */NXem_ebsd_crystal_structure_model/relative_intensity-field*
- */NXem_ebsd_crystal_structure_model/space_group-field*
- */NXem_ebsd_crystal_structure_model/unit_cell_abc-field*
- */NXem_ebsd_crystal_structure_model/unit_cell_alphabetagamma-field*
- */NXem_ebsd_crystal_structure_model/unit_cell_class-field*
- */NXem_ebsd_crystal_structure_model/unit_cell_volume-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXem_ebsd_crystal_structure_model.nxdl.xml

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*

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 2xm 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*
- */NXenergydispersion/tof_distance-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 are working much longer at the microscope, recalibrate and take multiple data items (scans, images, spectra). Each item comes with own detector and eventually on-the-fly processing settings and calibrations.

For the single data item use case one may argue that the need for an additional grouping is redundant. Instead, the metadata could equally be stored inside the respective groups of the top-level mandatory NX-instrument 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 or apertures used and/or changed. 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 having 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 pick a different value and continue. 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 (hardware components) and objects within the region-of-interest. Eventually external stimuli are applied and the specimen repositioned.

Some snapshot or integrated data from this stream are relevant for understanding signal genesis and electron/ion-beam-sample interaction (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).

Therefore, both the instrument and specimen should always 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, or *too poor* and thus either abort their session or try to bring the microscope first into a state where conditions are considered more stable, better, or of some whatever high enough quality to reuse data collection.

In all these cases it is practical to have a mechanism how time-dependent data of the instrument state can be stored and documented in a interoperable way. Where should these data be stored? With NeXus these data should not only be stored in the respective instrument component groups of the top-level NXinstrument. The reason is that this group should be reserved for as stable as possible quantities which do not change over the session. Thereby we can avoid to store repetitively that there is a certain gun or detector available but just store the changes. This is exactly what the em_lab group is for inside NXevent_data_em.

Ideally, NXevent_data_em are equipped with a start_time and end_time to represent a time interval (remind the idea of the instrument state stream) during which the scientist considered (or practically has to consider) the microscope (especially ebeam and specimen) as stable enough.

Arguably it is oftentimes tricky to specify a clear time interval when the microscope is stable enough. Take for instance the acquisition of an image or spectra stack. It is not fully possible (technically) to avoid that even within a single image instabilities of the beam are faced and drift occurs. Maybe in many cases this these instabilities are irrelevant but does this warrant to create a data schema where either the microscope state can only be stored very coarsely or one is forced to store it very finely?

This is a question of how one wishes to granularize pieces of information. A possible solution is to consider each probed position, i.e. point in time when the beam was not blanked and thus when the beam illuminates a portion of the material, i.e. the interaction volume, whose signal contributions are then counted by the one or multiple detector(s) as per pixel- or per voxel signal in the region-of-interest. NXevent_data_em in combination with the NXem application definition allows researchers to document this. Noteworty to mention is that we understand that in many cases realizing such a fine temporal and logical granularization and data collection is hard to achieve in practice.

A frequently made choice, mainly for convenience, is that drift and scan distortions are considered a feature or inaccuracy of the image and/or spectrum and thus one de facto accepts that the microscope was not as stable as expected during the acquisition of the image. We learn that the idea of a time interval during the microscope session may be interpreted differently by different users. Here we consider the choice to focus on images and spectra, and eventually single position measurements as the smallest granularization level. Which eventually may require to add optional NXprocess instances for respectively collected data to describe the relevant distortions. Nevertheless, the distortions are typically corrected for by numerical protocols. This fact warrants to consider the distortion correction a computational workflow which can be modelled as a chain of NXprocess instances each with own parameters. an own A more detailed overview of such computational steps to cope with scan distortions is available in the literature:

- C. Ophus et al.
- B. Berkels et al.
- L. Jones et al.

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 axi; 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 offspring application definition of NXem which documents metadata for tracking explicitly electrons (with ray-tracing based descriptors/computational geometry descriptors) or tracking of wave bundles.

This perspective on much 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 a microscope.

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 additional groups of the NXinstrument though inside instances of here detailed NXevent_data_em.

Symbols:

No symbol table

Groups cited:

NXimage_set, *NXinstrument*, *NXinteraction_vol_em*, *NXspectrum_set*, *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 information included when the snapshot time interval ended.

event_identifier: (optional) *NX_CHAR*

Reference to a specific state and setting of the microscope.

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.

This field may also be used for storing additional information about the event.

IMAGE_SET: (optional) *NXimage_set*

SPECTRUM_SET: (optional) *NXspectrum_set*

INSTRUMENT: (optional) *NXinstrument*

USER: (optional) *NXuser*

INTERACTION_VOL_EM: (optional) *NXinteraction_vol_em*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXevent_data_em/end_time-field*
- */NXevent_data_em/event_identifier-field*
- */NXevent_data_em/event_type-field*
- */NXevent_data_em/IMAGE_SET-group*
- */NXevent_data_em/INSTRUMENT-group*
- */NXevent_data_em/INTERACTION_VOL_EM-group*
- */NXevent_data_em/SPECTRUM_SET-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*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXevent_data_em_set.nxdl.xml

NXfabrication

Status:

base class, extends *NXobject*

Description:

Details about a component as defined by its manufacturer.

Symbols:

No symbol table

Groups cited:
none

Structure:

vendor: (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.

- */NXfabrication/capability-field*
- */NXfabrication/identifier-field*
- */NXfabrication/model-field*
- */NXfabrication/vendor-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXfabrication.nxdl.xml

NXfiber

Status:

base class, extends *NXobject*

Description:

An optical fiber, e.g. glass fiber.

Specify the quantities that define the fiber. Fiber optics are described in detail [here](https://www.photonics.com/Article.aspx?AID=25151&PID=4), for example.

Symbols:

N_spectrum_core: Length of the spectrum vector (e.g. wavelength or energy) for which the refractive index of the core material is given.

N_spectrum_clad: Length of the spectrum vector (e.g. wavelength or energy) for which the refractive index of the cladding material is given.

N_spectrum_attenuation: Length of the spectrum vector (e.g. wavelength or energy) for which the attenuation curve is given.

Groups cited:
NXsample

Structure:

description: (recommended) *NX_CHAR*

Descriptive name or brief description of the fiber, e.g. by stating its dimension. The dimension of a fiber can be given as 60/100/200 which refers to a core diameter of 60 micron, a clad diameter of 100 micron, and a coating diameter of 200 micron.

type: (optional) *NX_CHAR*

Type/mode of the fiber. Modes of fiber transmission are shown in Fig. 5 [here](https://www.photonics.com/Article.aspx?AID=25151&PID=4).

Any of these values:

- `single mode`
- `multimode graded index`

- `multimode step index`

dispersion_type: (optional) *NX_CHAR*

Type of dispersion.

Any of these values: `modal` | `material` | `chromatic`

dispersion: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [N_spectrum_core]) {units=*NX_TIME*}

Spectrum-dependent (or refractive index-dependent) dispersion of the fiber. Specify in ps/nm*km.

length: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Length of the fiber.

spectral_range: (recommended) *NX_FLOAT* (Rank: 1, Dimensions: [2]) {units=*NX_ANY*}

Spectral range for which the fiber is designed. Enter the minimum and maximum values (lower and upper limit) of the wavelength range. **@units**: (optional) *NX_CHAR*

Unit of spectral array (e.g. nanometer or angstrom for wavelength, or electronvolt for energy etc.).

transfer_rate: (optional) *NX_FLOAT* {units=*NX_ANY*}

Transfer rate of the fiber (in GB per second).

@units: (optional) *NX_CHAR*

GB/s

numerical_aperture: (optional) *NX_FLOAT* {units=*NX_UNITLESS*}

Numerical aperture (NA) of the fiber.

attenuation: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [N_spectrum_attenuation]) {units=*NX_ANY*}

Wavelength-dependent attenuation of the fiber (specify in dB/km).

@units: (optional) *NX_CHAR*

Use dB/km.

Obligatory value: `dB/km`

power_loss: (optional) *NX_FLOAT* {units=*NX_UNITLESS*}

Power loss of the fiber in percentage.

acceptance_angle: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

Acceptance angle of the fiber.

core: (optional) *NXsample*

Core of the fiber, i.e. the part of the fiber which transmits the light.

core_material: (optional) *NX_CHAR*

Specify the material of the core of the fiber.

core_diameter: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Core diameter of the fiber (e.g. given in micrometer).

core_index_of_refraction: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [2, N_spectrum_core]) {units=*NX_UNITLESS*}

Complex index of refraction of the fiber. Specify at given wavelength (or energy, wavenumber etc.) values.

cladding: (optional) *NXsample*

Core of the fiber, i.e. the part of the fiber which transmits the light.

clad_material: (optional) *NX_CHAR*

Specify the material of the core of the fiber.

clad_diameter: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Clad diameter of the fiber (e.g. given in micrometer).

clad_index_of_refraction: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [2, N_spectrum_clad]) {units=*NX_UNITLESS*}

Complex index of refraction of the fiber. Specify at given wavelength (or energy, wavenumber etc.) values.

coating: (optional) *NXsample*

Coating of the fiber.

coating_material: (optional) *NX_CHAR*

Specify the material of the coating of the fiber.

coating_diameter: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Outer diameter of the fiber (e.g. given in micrometer).

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXfiber/acceptance_angle-field*
- */NXfiber/attenuation-field*
- */NXfiber/attenuation@units-attribute*
- */NXfiber/cladding-group*
- */NXfiber/cladding/clad_diameter-field*
- */NXfiber/cladding/clad_index_of_refraction-field*
- */NXfiber/cladding/clad_material-field*
- */NXfiber/coating-group*
- */NXfiber/coating/coating_diameter-field*
- */NXfiber/coating/coating_material-field*
- */NXfiber/core-group*
- */NXfiber/core/core_diameter-field*
- */NXfiber/core/core_index_of_refraction-field*
- */NXfiber/core/core_material-field*
- */NXfiber/description-field*
- */NXfiber/dispersion-field*

- */NXfiber/dispersion_type-field*
- */NXfiber/length-field*
- */NXfiber/numerical_aperture-field*
- */NXfiber/power_loss-field*
- */NXfiber/spectral_range-field*
- */NXfiber/spectral_range@units-attribute*
- */NXfiber/transfer_rate-field*
- */NXfiber/transfer_rate@units-attribute*
- */NXfiber/type-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXfiber.nxdl.xml

NXgraph_edge_set

Status:

base class, extends *NXobject*

Description:

A set of (eventually directed) edges which connect nodes/vertices of a graph.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_edges: The number of edges.

Groups cited:
none

Structure:

number_of_edges: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Total number of edges, counting eventual bidirectional edges only once.

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing edges. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [n_edges]) {units=*NX_UNITLESS*}

Integer used to distinguish edges for explicit indexing.

directionality: (optional) *NX_INT* (Rank: 1, Dimensions: [n_edges]) {units=*NX_UNITLESS*}

Specifier whether each edge is non-directional, one-directional, or bidirectional. Use the smallest available binary representation which can store three different states:

- 0 / state 0x00 is non-directional

- 1 / state 0x01 is one-directional
- 2 / state 0x02 is bi-directional

node_pair: (optional) *NX_INT* (Rank: 2, Dimensions: [n_edges, 2]) {units=*NX_UNITLESS*}

Pairs of node/vertex identifier. Each pair represents the connection between two nodes.

In the case that the edge is non- or bi-directional node identifier should be stored in ascending order is preferred.

In the case of one-directional, for each pair the identifier of the source node is the first entry in the pair. The identifier of the target is the second entry in the pair, i.e. the pair encodes the information as if one traverses the edge from the source node walking to the target node.

is_a: (optional) *NX_CHAR* (Rank: 1, Dimensions: [c])

A human-readable qualifier which type or e.g. class instance the edge is an instance of.

label: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_edges])

A human-readable label/caption/tag for the edge.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXgraph_edge_set/directionality-field*
- */NXgraph_edge_set/identifier-field*
- */NXgraph_edge_set/identifier_offset-field*
- */NXgraph_edge_set/is_a-field*
- */NXgraph_edge_set/label-field*
- */NXgraph_edge_set/node_pair-field*
- */NXgraph_edge_set/number_of_edges-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXgraph_edge_set.nxdl.xml

NXgraph_node_set

Status:

base class, extends *NXobject*

Description:

A set of nodes/vertices in space representing members of a graph.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: The dimensionality of the graph. Eventually use one for categorical.

c: The cardinality of the set, i.e. the number of nodes/vertices of the graph.

Groups cited:
none

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing nodes. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Integer used to distinguish nodes for explicit indexing.

is_a: (optional) *NX_CHAR* (Rank: 1, Dimensions: [c])

A human-readable qualifier which type or e.g. class instance the node is an instance of. As e.g. a NeXus application definition is a graph, more specifically a hierarchical directed labelled property graph, instances which are groups like NXgraph_node_set could have an is_a qualifier reading NXgraph_node_set.

label: (optional) *NX_CHAR* (Rank: 1, Dimensions: [c])

A human-readable label/caption/tag of the graph.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXgraph_node_set/cardinality-field*
- */NXgraph_node_set/dimensionality-field*
- */NXgraph_node_set/identifier-field*
- */NXgraph_node_set/identifier_offset-field*
- */NXgraph_node_set/is_a-field*
- */NXgraph_node_set/label-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXgraph_node_set.nxdl.xml

NXgraph_root

Status:

base class, extends *NXobject*

Description:

An instance of a graph.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXgraph_edge_set, *NXgraph_node_set*

Structure:

nodes: (optional) *NXgraph_node_set*

relation: (optional) *NXgraph_edge_set*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXgraph_root/nodes-group*
- */NXgraph_root/relation-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXgraph_root.nxdl.xml

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*, *NXfabrication*, *NXion*, *NXlens_em*, *NXsensor*, *NXsource*, *NXtransformations*

Structure:

FABRICATION: (optional) *NXfabrication*

ion_source: (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/FABRICATION-group*
- */NXibeam_column/ion_source-group*
- */NXibeam_column/ion_source/brightness-field*
- */NXibeam_column/ion_source/current-field*
- */NXibeam_column/ion_source/description-field*
- */NXibeam_column/ion_source/emitter_type-field*
- */NXibeam_column/ion_source/ion_energy_profile-field*
- */NXibeam_column/ion_source/name-field*
- */NXibeam_column/ion_source/probe-group*
- */NXibeam_column/ion_source/TRANSFORMATIONS-group*
- */NXibeam_column/ion_source/voltage-field*
- */NXibeam_column/LENS_EM-group*
- */NXibeam_column/SENSOR-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXibeam_column.nxdl.xml

NXimage_set

Status:

base class, extends *NXobject*

Description:

Container for reporting a set of images taken.

Symbols:

n_images: Number of images in the stack.

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*, *NXprogram*

Structure:

PROCESS: (optional) *NXprocess*

Details how images were processed from the detector readings.

source: (optional) *NX_CHAR*

Resolvable data artifact (e.g. filename) from which the all values in the NXdata instances in this NXimage_set were loaded during parsing. **@version**: (optional) *NX_CHAR*

An at least as strong as SHA256 hashvalue of the data artifact which source represents digitally to support provenance tracking.

mode: (optional) *NX_CHAR*

Imaging (data collection) mode of the instrument during acquisition of the data in this NXimage_set instance.

detector_identifier: (optional) *NX_CHAR*

Link or name of an NXdetector instance with which the data were collected.

PROGRAM: (optional) *NXprogram*

stack: (optional) *NXdata*

Image (stack).

data_counts: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [n_images, n_y, n_x]) {units=*NX_UNITLESS*} *<=*

Image intensity values.

axis_image_identifier: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_images]) {units=*NX_UNITLESS*}

Image identifier

@long_name: (optional) *NX_CHAR*

Image identifier.

axis_y: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

Pixel coordinate center of mass along y direction.

@long_name: (optional) *NX_CHAR*

Coordinate along y direction.

axis_x: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_LENGTH*}

Pixel coordinate center of mass along x direction.

@long_name: (optional) *NX_CHAR*

Coordinate along x direction.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXimage_set/PROCESS-group*
- */NXimage_set/PROCESS/detector_identifier-field*
- */NXimage_set/PROCESS/mode-field*
- */NXimage_set/PROCESS/PROGRAM-group*
- */NXimage_set/PROCESS/source-field*
- */NXimage_set/PROCESS/source@version-attribute*

- */NXimage_set/stack-group*
- */NXimage_set/stack/axis_image_identifier-field*
- */NXimage_set/stack/axis_image_identifier@long_name-attribute*
- */NXimage_set/stack/axis_x-field*
- */NXimage_set/stack/axis_x@long_name-attribute*
- */NXimage_set/stack/axis_y-field*
- */NXimage_set/stack/axis_y@long_name-attribute*
- */NXimage_set/stack/data_counts-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXimage_set.nxdl.xml

NXimage_set_em_adf

Status:

base class, extends *NXobject*

Description:

Container for reporting a set of annular dark field images.

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 for which the settings, binning, and energy resolution is the same for each scan point.

Symbols:

n_images: Number of images in the stack.

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 how (HA)ADF images were processed from the detector readings.

source: (optional) *NX_CHAR*

Typically the name of the input, (vendor) file from which all the NXdata instances in this NXimage_set_em_adf were loaded during parsing to represent them in e.g. databases. **@version**: (optional) *NX_CHAR*

An at least as strong as SHA256 hashvalue of the dataset/file which represents the source digitally to support provenance tracking.

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

stack: (optional) *NXdata*

Annular dark field image stack.

data_counts: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [n_images, n_y, n_x]) {units=*NX_UNITLESS*} *<=*

Image intensity values.

axis_image_identifier: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_images]) {units=*NX_UNITLESS*}

Image identifier

@long_name: (optional) *NX_CHAR*

Image ID.

axis_y: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

Pixel center of mass along y direction.

@long_name: (optional) *NX_CHAR*

Coordinate along y direction.

axis_x: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_LENGTH*}

Pixel center of mass along x direction.

@long_name: (optional) *NX_CHAR*

Coordinate along x direction.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */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*
- */NXimage_set_em_adf/PROCESS/source-field*
- */NXimage_set_em_adf/PROCESS/source@version-attribute*
- */NXimage_set_em_adf/stack-group*

- */NXimage_set_em_adf/stack/axis_image_identifier-field*
- */NXimage_set_em_adf/stack/axis_image_identifier@long_name-attribute*
- */NXimage_set_em_adf/stack/axis_x-field*
- */NXimage_set_em_adf/stack/axis_x@long_name-attribute*
- */NXimage_set_em_adf/stack/axis_y-field*
- */NXimage_set_em_adf/stack/axis_y@long_name-attribute*
- */NXimage_set_em_adf/stack/data_counts-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXimage_set_em_adf.nxdl.xml

NXimage_set_em_kikuchi

Status:

base class, extends *NXobject*

Description:

Measured set of electron backscatter diffraction data, aka Kikuchi pattern. Kikuchi pattern are the raw data for computational workflows in the field of orientation (imaging) microscopy. The technique is especially used 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 before a respective digital model of the inspected microstructure can be analyzed. The resulting volume is often termed a so-called (representative) volume element (RVE). Several software packages are available for performing this post-processing. For now we do not consider metadata of these post-processing steps as a part of this base class because the connection between the large variety of such post-processing steps and the measured electron microscopy data is usually very small.

If we envision a (knowledge) graph for EBSD it consists of individual sub-graphs which convey information about the specimen preparation, the measurement of the specimen in the electron microscope, the indexing of the collected Kikuchi pattern stack, eventual post-processing of the indexed orientation image via similarity grouping algorithms to yield (grains, texture). Conceptually these post-processing steps are most frequently serving the idea to reconstruct quantitatively so-called microstructural features (grains, phases, interfaces). Materials scientists use these features according to the multi-scale materials modeling paradigm to infer material properties. They do so by quantifying correlations between the spatial arrangement of the features, their individual properties, and (macroscopic) properties of materials.

Symbols:

n_sc: Number of scanned points. Scan point may have none, one, or more pattern.

n_p: Number of diffraction pattern.

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:
NXdata, *NXprocess*

Structure:

PROCESS: (optional) *NXprocess*

Details how Kikuchi pattern were processed from the detector readings. Scientists interested in EBSD should inspect the respective NXem_ebsd application definition which can be used as a partner application definition to detail substantially more details to this processing.

stack: (optional) *NXdata*

Collected Kikuchi pattern as an image stack. As raw and closest to the first retrievable measured data as possible, i.e. do not use this container to store already averaged, filtered or whatever post-processed pattern unless these are generated unmodifiably by the instrument given the way how the instrument and control software was configured for your microscope session. **scan_point_identifier**: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_p]) {units=*NX_UNITLESS*}

Array which resolves the scan point to which each pattern belongs. Scan points are evaluated in sequence starting from scan point zero until scan point n_sc - 1. Evaluating the cumulated of this array decodes which pattern in intensity belong to which scan point. In an example we may assume we collected three scan points. For the first we measure one pattern, for the second we measure three pattern, for the last we measure no pattern. The values of scan_point_identifier will be 0, 1, 1, 1, as we have measured four pattern in total.

In most cases usually one pattern is averaged by the detector for some amount of time and then reported as one pattern. Use compressed arrays allows to store the scan_point_identifier efficiently.

data_counts: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [n_p, n_y, n_x]) {units=*NX_UNITLESS*} *<=*

Signal intensity. For so-called three-dimensional or serial sectioning EBSD it is necessary to follow a sequence of specimen surface preparation and data collection. In this case users should collect the data for each serial sectioning step in an own instance of NXimage_set_em_kikuchi. All eventual post-processing of these measured data should be documented via NXebsd, resulting microstructure representations should be stored as NXms. **@long_name**: (optional) *NX_CHAR* *<=*

Kikuchi pattern intensity

pattern_identifier: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_p]) {units=*NX_UNITLESS*}

Pattern are enumerated starting from 0 to n_p - 1.

@long_name: (optional) *NX_CHAR*

Kikuchi pattern identifier

axis_y: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

Pixel coordinate along the y direction.

@long_name: (optional) *NX_CHAR*

Label for the y axis

axis_x: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_LENGTH*}

Pixel coordinate along the x direction.

@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_kikuchi/PROCESS-group*
- */NXimage_set_em_kikuchi/stack-group*
- */NXimage_set_em_kikuchi/stack/axis_x-field*
- */NXimage_set_em_kikuchi/stack/axis_x@long_name-attribute*
- */NXimage_set_em_kikuchi/stack/axis_y-field*
- */NXimage_set_em_kikuchi/stack/axis_y@long_name-attribute*
- */NXimage_set_em_kikuchi/stack/data_counts-field*
- */NXimage_set_em_kikuchi/stack/data_counts@long_name-attribute*
- */NXimage_set_em_kikuchi/stack/pattern_identifier-field*
- */NXimage_set_em_kikuchi/stack/pattern_identifier@long_name-attribute*
- */NXimage_set_em_kikuchi/stack/scan_point_identifier-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXimage_set_em_kikuchi.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:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXinteraction_vol_em.nxdl.xml

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:

identifier: (optional) *NX_CHAR*

A unique identifier whereby such an ion can be referred to via the service offered as described in identifier_type.

identifier_type: (optional) *NX_CHAR*

How can the identifier be resolved?

Obligatory value: `inchi`

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: 2, Dimensions: [1, 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:

The hash value H is $H = Z + N * 256$ 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)

nuclid_list: (optional) *NX_UINT* (Rank: 2, Dimensions: [2, n_ivecmax]) {units=*NX_UNITLESS*}

A supplementary row vector which decodes the isotope_vector into a human-readable matrix of nuclids with the following formatting:

The first row specifies the isotope mass number, i.e. using the hashvalues from the isotope_vector this is $Z + N$. As an example for a carbon-14 isotope the number is 14. The second row specifies the number of protons Z, e.g. 6 for the carbon-14 example. This row matrix is thus a mapping the notation of using superscribed isotope mass and subscripted number of protons to identify isotopes. Unused places filling up to n_ivecmax need to be filled with zero.

color: (optional) *NX_CHAR*

Color code used for visualizing such ions.

volume: (optional) *NX_FLOAT* {units=*NX_VOLUME*}

Assumed volume of the ion.

In atom probe microscopy this field can be used to store the reconstructed volume per ion (average) which is typically stored in range files and will be used when building a tomographic reconstruction of an atom probe dataset.

charge: (optional) *NX_FLOAT* {units=*NX_CHARGE*}

Charge of the ion.

charge_state: (optional) *NX_INT* {units=*NX_UNITLESS*}

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, for the field of atom probe microscopy 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 (mqmin) and upper (mqmax) bounds of mass-to-charge-state ratio interval(s) [mqmin, mqmax] (boundaries included) for which the respective ion is one to be labelled with ion_identifier. The field is primarily of interest to document the result of indexing a ToF/mass spectrum.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXion/charge-field*
- */NXion/charge_state-field*
- */NXion/color-field*
- */NXion/identifier-field*
- */NXion/identifier_type-field*
- */NXion/ion_type-field*
- */NXion/isotope_vector-field*
- */NXion/mass_to_charge_range-field*
- */NXion/name-field*
- */NXion/nuclid_list-field*
- */NXion/volume-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXion.nxdl.xml

NXisocontour

Status:

base class, extends *NXobject*

Description:

Computational geometry description of isocontouring/phase-fields in Euclidean space.

Iso-contouring algorithms such as MarchingCubes and others are frequently used to segment d-dimensional regions into regions where intensities are lower or higher than a threshold value, the so-called isovalue.

Frequently in computational materials science phase-field methods are used which generate data on discretized grids. Isocontour algorithms are often used in such context to pinpoint the locations of microstructural features from this implicit phase-field-variable-based description.

One of the key intentions of this base class is to provide a starting point for scientists from the phase-field community (condensed matter physicists, and materials engineers) to incentivize that also phase-field simulation data could be described with NeXus, provided base classes such as the this one get further extend according to the liking of the phase-field community.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: The dimensionality of the description.

Groups cited:
NXcg_grid

Structure:

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

isovalue: (optional) *NX_NUMBER* {units=*NX_ANY*}

The threshold or iso-contour value.

grid: (optional) *NXcg_grid*

The discretized grid on which the iso-contour algorithm operates.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXisocontour/dimensionality-field*
- */NXisocontour/grid-group*
- */NXisocontour/isovalue-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXisocontour.nxdl.xml

NXiv_temp

Status:

application definition, extends *NXsensor_scan*

Description:

Application definition for temperature-dependent IV curve measurements.

In this application definition, times should be specified always together with an UTC offset.

This is the application definition describing temperature dependent IV curve measurements. For this a temperature is set. After reaching the temperature, a voltage sweep is performed. For each voltage a current is measured. Then, the next desired temperature is set and an IV measurement is performed.

Symbols:

n_different_temperatures: Number of different temperature setpoints used in the experiment.

n_different_voltages: Number of different voltage setpoints used in the experiment.

Groups cited:
NXdata, *NXentry*, *NXenvironment*, *NXinstrument*, *NXsensor*

Structure:

ENTRY: (required) *NXentry <=*

definition: (required) *NX_CHAR <=*

Obligatory value: `NXiv_temp`

INSTRUMENT: (required) *NXinstrument <=*

ENVIRONMENT: (required) *NXenvironment <=*

Describes an environment setup for a temperature-dependent IV measurement experiment.

The temperature and voltage must be present as independently scanned controllers and the current sensor must also be present with its readings. **voltage_controller**: (required) *NXsensor <=*

temperature_controller: (required) *NXsensor <=*

current_sensor: (required) *NXsensor <=*

DATA: (required) *NXdata <=*

This NXdata should contain separate fields for the current values at different temperature setpoints, for example current_at_100C. There should also be two more fields called temperature and voltage containing the setpoint values. There should also be a field with an array of rank equal to the number of different temperature setpoints and each child's dimension equal to the number of voltage setpoints. **temperature**: (required) *NX_NUMBER*

voltage: (required) *NX_NUMBER*

current: (required) *NX_NUMBER* (Rank: 2, Dimensions: [n_different_temperatures, n_different_voltages])

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXiv_temp/ENTRY-group*
- */NXiv_temp/ENTRY/DATA-group*
- */NXiv_temp/ENTRY/DATA/current-field*
- */NXiv_temp/ENTRY/DATA/temperature-field*
- */NXiv_temp/ENTRY/DATA/voltage-field*
- */NXiv_temp/ENTRY/definition-field*
- */NXiv_temp/ENTRY/INSTRUMENT-group*
- */NXiv_temp/ENTRY/INSTRUMENT/ENVIRONMENT-group*
- */NXiv_temp/ENTRY/INSTRUMENT/ENVIRONMENT/current_sensor-group*
- */NXiv_temp/ENTRY/INSTRUMENT/ENVIRONMENT/temperature_controller-group*
- */NXiv_temp/ENTRY/INSTRUMENT/ENVIRONMENT/voltage_controller-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXiv_temp.nxdl.xml

NXlab_electro_chemo_mechanical_preparation

Status:

application definition, extends *NXobject*

Description:

Grinding and polishing of a sample using abrasives in a wet lab. Manual procedures, electro-chemical, vibropolishing.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXentry, *NXfabrication*, *NXprocess*, *NXsample*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXlab_electro_chemo_mechanical_preparation`

workflow_step_identifier: (required) *NX_UINT*

workflow_step_description: (required) *NX_CHAR*

SAMPLE: (required) *NXsample <=*

USER: (required) *NXuser <=*

grinding_machine: (required) *NXfabrication*

vendor: (required) *NX_CHAR <=*

model: (required) *NX_CHAR <=*

identifier: (recommended) *NX_CHAR <=*

GRINDING_STEP: (required) *NXprocess <=*

A preparation step performed by a human or a robot/automated system.

sequence_index: (required) *NX_POSINT <=*

start_time: (required) *NX_DATE_TIME*

end_time: (required) *NX_DATE_TIME*

abrasive_medium_carrier: (required) *NX_CHAR*

Carrier/plate used on which the abrasive/(lubricant) mixture was applied.

abrasive_medium: (required) *NX_CHAR*

Medium on the abrasive_medium_carrier (cloth or grinding plate) whereby material is abrasively weared.

lubricant: (required) *NX_CHAR*

Lubricant

rotation_control: (required) *NX_CHAR*

Qualitative statement how the revelation of the machine was configured. If the rotation was controlled manually, e.g. by turning knobs choose manual and estimate the nominal average rotation. If the rotation was controlled via choosing from a fixed set of options offered by the machine choose fixed and specify the nominal rotation. If programmed use rotation_history (e.g. for automated/robot systems).

Any of these values: `undefined` | `manual` | `fixed` | `programmed`

force_control: (required) *NX_CHAR*

Qualitative statement how the (piston) force with which the sample was pressed into/against the abrasive medium was controlled if at all. If the force was controlled manually e.g. by turning knobs choose manual and estimate nominal average force. If the force was controlled via choosing from a fixed set of options offered by the machine choose fixed and specify the nominal force. If programmed use force_history (e.g. for automated/robot systems).

Any of these values: `undefined` | `manual` | `fixed` | `programmed`

time_control: (required) *NX_CHAR*

Qualitative statement for how long (assuming regular uninterrupted) preparation at the specified conditions the preparation step was applied.

Any of these values: `undefined` | `manual` | `fixed` | `programmed`

rotation: (required) *NX_NUMBER* {units=*NX_FREQUENCY*}

Turns per unit time.

force: (required) *NX_NUMBER* {units=*NX_ANY*}

Force exerted on the sample to press it into the abrasive.

time: (required) *NX_NUMBER* {units=*NX_TIME*}

Seconds

removal: (required) *NX_CHAR*

Qualitative statement how the material removal was characterized.

Any of these values: `undefined` | `estimated` | `measured`

thickness_reduction: (required) *NX_NUMBER* {units=*NX_LENGTH*}

How thick a layer was removed.

CLEANING_STEP: (required) *NXprocess* <=

A preparation step performed by a human or a robot/automated system with the aim to remove residual abrasive medium from the specimen surface. **sequence_index**: (required) *NX_POSINT* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXlab_electro_chemo_mechanical_preparation/ENTRY-group*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/CLEANING_STEP-group*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/CLEANING_STEP/sequence_index-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/definition-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/grinding_machine-group*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/grinding_machine/identifier-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/grinding_machine/model-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/grinding_machine/vendor-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP-group*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP/abrasive_medium-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP/abrasive_medium_carrier-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP/end_time-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP/force-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP/force_control-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP/lubricant-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP/removal-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP/rotation-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP/rotation_control-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP/sequence_index-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP/start_time-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP/thickness_reduction-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP/time-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/GRINDING_STEP/time_control-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/SAMPLE-group*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/USER-group*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/workflow_step_description-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY/workflow_step_identifier-field*
- */NXlab_electro_chemo_mechanical_preparation/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXlab_electro_chemo_mechanical_preparation.nxdl.xml

NXlab_sample_mounting

Status:

application definition, extends *NXobject*

Description:

Embedding of a sample in a medium for easing processability.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXentry, *NXfabrication*, *NXsample*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXlab_sample_mounting`

start_time: (required) *NX_DATE_TIME* <=

end_time: (required) *NX_DATE_TIME* <=

mounting_method: (required) *NX_CHAR*

Qualitative statement how the sample was mounted.

Any of these values: `cold_mounting` | `hot_mounting`

embedding_medium: (required) *NX_CHAR*

Type of material.

Any of these values: `resin` | `epoxy`

electrical_conductivity: (required) *NX_FLOAT* {units=*NX_ANY*}

Electrical conductivity of the embedding medium.

SAMPLE: (required) *NXsample* <=

USER: (required) *NXuser* <=

mounting_machine: (required) *NXfabrication*

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXlab_sample_mounting/ENTRY-group*
- */NXlab_sample_mounting/ENTRY/definition-field*
- */NXlab_sample_mounting/ENTRY/electrical_conductivity-field*
- */NXlab_sample_mounting/ENTRY/embedding_medium-field*
- */NXlab_sample_mounting/ENTRY/end_time-field*
- */NXlab_sample_mounting/ENTRY/mounting_machine-group*
- */NXlab_sample_mounting/ENTRY/mounting_machine/identifier-field*
- */NXlab_sample_mounting/ENTRY/mounting_machine/model-field*
- */NXlab_sample_mounting/ENTRY/mounting_machine/vendor-field*
- */NXlab_sample_mounting/ENTRY/mounting_method-field*
- */NXlab_sample_mounting/ENTRY/SAMPLE-group*
- */NXlab_sample_mounting/ENTRY/start_time-field*
- */NXlab_sample_mounting/ENTRY/USER-group*
- */NXlab_sample_mounting/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXlab_sample_mounting.nxdl.xml

NXlens_em

Status:

base class, extends *NXobject*

Description:

Description of an electro-magnetic lens or a compound lens.

For NXtransformations 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 NXtransformations.

For details of electro-magnetic lenses in the literature see e.g. L. Reimer

Symbols:

No symbol table

Groups cited:
NXfabrication, *NXtransformations*

Structure:

type: (optional) *NX_CHAR*

Qualitative type of lens with respect to the number of pole pieces.

Any of these values:

- `single`

- `double`
- `quadrupole`
- `hexapole`
- `octupole`

name: (optional) *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.

value: (optional) *NX_NUMBER* {units=*NX_ANY*}

This field should be used when the exact voltage or current of the lens is not directly controllable as the control software of the microscope does not enable users/or is was not configured to enable the user to retrieve these values. In this case this field should be used to specify the value as read from the control software. Although consumers of the application definition should not expect this value to represent the exact physical voltage or excitation, it is still useful to know though as it allows other users to reuse this lens setting, which, provided a properly working instrument and software should bring the lenses into a similar state.

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.

FABRICATION: (optional) *NXfabrication*

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/FABRICATION-group*
- */NXlens_em/manufacturer_name-field*
- */NXlens_em/model-field*
- */NXlens_em/name-field*
- */NXlens_em/TRANSFORMATIONS-group*
- */NXlens_em/type-field*
- */NXlens_em/value-field*
- */NXlens_em/voltage-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXlens_em.nxdl.xml

NXlens_opt

Status:

base class, extends *NXobject*

Description:

Description of an optical lens.

Symbols:

N_spectrum: Size of the wavelength array for which the refractive index of the material is given.

N_spectrum_coating: Size of the wavelength array for which the refractive index of the coating is given.

N_spectrum_RT: Size of the wavelength array for which the reflectance or transmission of the lens is given.

Groups cited:
NXsample

Structure:

type: (optional) *NX_CHAR*

Type of the lens (e.g. concave, convex etc.).

Any of these values:

- `biconcave`
- `plano-concave`
- `convexo-concave`
- `biconvex`

- `plano-convex`
- `concavo-convex`
- `Fresnel lens`
- `other`

other_type: (optional) *NX_CHAR*

If you chose 'other' as type specify what it is.

chromatic: (optional) *NX_BOOLEAN*

Is it a chromatic lens?

lens_diameter: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Diameter of the lens.

reflectance: (optional) *NX_CHAR* (Rank: 1, Dimensions: [N_spectrum_RT]) {units=*NX_UNITLESS*}

Reflectance of the lens at given spectral values.

transmission: (optional) *NX_CHAR* (Rank: 1, Dimensions: [N_spectrum_RT]) {units=*NX_UNITLESS*}

Transmission of the lens at given spectral values.

focal_length: (recommended) *NX_NUMBER* (Rank: 1, Dimensions: [2]) {units=*NX_LENGTH*}

Focal length of the lens on the front side (first value), i.e. where the beam is incident, and on the back side (second value).

curvature_radius_FACE: (recommended) *NX_NUMBER* {units=*NX_LENGTH*}

Curvature radius of the lens. Instead of 'FACE' in the name of this field, the user is advised to specify for which surface (e.g. front or back) the curvature is provided: e.g. curvature_front or curvature_back. The front face is the surface on which the light beam is incident, while the back face is the one from which the light beam exits the lens.

Abbe_number: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

Abbe number (or V-number) of the lens.

substrate: (optional) *NXsample*

Properties of the substrate material of the lens. If the lens has a coating specify the coating material and its properties in 'coating'. **substrate_material**: (optional) *NX_CHAR*

Specify the substrate material of the lens.

substrate_thickness: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Thickness of the lens substrate at the optical axis.

index_of_refraction: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [2, N_spectrum]) {units=*NX_UNITLESS*}

Complex index of refraction of the lens material. Specify at given wavelength (or energy, wavenumber etc.) values.

COATING: (optional) *NXsample*

If the lens has a coating describe the material and its properties. Some basic information can be found e.g. [here] (https://www.opto-e.com/basics/reflection-transmission-and-coatings). If the back and front side of the lens are coated with different materials, use separate COATING(NXsample) fields to describe the coatings on the front and back side, respectively. For example: coating_front(NXsample) and coating_back(NXsample). **coating_type**: (optional) *NX_CHAR*

Specify the coating type (e.g. dielectric, anti-reflection (AR), multilayer coating etc.).

coating_material: (optional) *NX_CHAR*

Describe the coating material (e.g. MgF2).

coating_thickness: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Thickness of the coating.

index_of_refraction_coating: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [2, N_spectrum_coating]) {units=*NX_UNITLESS*}

Complex index of refraction of the coating. Specify at given spectral values (wavelength, energy, wavenumber etc.).

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXlens_opt/Abbe_number-field*
- */NXlens_opt/chromatic-field*
- */NXlens_opt/COATING-group*
- */NXlens_opt/COATING/coating_material-field*
- */NXlens_opt/COATING/coating_thickness-field*
- */NXlens_opt/COATING/coating_type-field*
- */NXlens_opt/COATING/index_of_refraction_coating-field*
- */NXlens_opt/curvature_radius_FACE-field*
- */NXlens_opt/focal_length-field*
- */NXlens_opt/lens_diameter-field*
- */NXlens_opt/other_type-field*
- */NXlens_opt/reflectance-field*
- */NXlens_opt/substrate-group*
- */NXlens_opt/substrate/index_of_refraction-field*
- */NXlens_opt/substrate/substrate_material-field*
- */NXlens_opt/substrate/substrate_thickness-field*
- */NXlens_opt/transmission-field*
- */NXlens_opt/type-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXlens_opt.nxdl.xml

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 trespect 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 positon 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

NXmatch_filter

Status:

base class, extends *NXobject*

Description:

Settings of a filter to select or remove entries based on their value.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_values: How many different match values does the filter specify.

Groups cited:
none

Structure:

method: (optional) *NX_CHAR*

Meaning of the filter: Whitelist specifies which entries with said value to include. Entries with all other values will be filtered out.

Blacklist specifies which entries with said value to exclude. Entries with all other values will be included.

Any of these values: `whitelist` | `blacklist`

match: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_values]) {units=*NX_UNITLESS*}

Array of values to filter according to method. For example if the filter specifies [1, 5, 6] and method is whitelist, only entries with values matching 1, 5 or 6 will be processed. All other entries will be filtered out.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXmatch_filter/match-field*
- */NXmatch_filter/method-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmatch_filter.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*, *NXuser*

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 <=*

USER: (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: (recommended) *NX_CHAR <=*

Name of the affiliation of the user at the point in time when the experiment was performed.

address: (recommended) *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/.

INSTRUMENT: (required) *NXinstrument <=*

energy_resolution: (required) *NX_FLOAT* {units=*NX_ENERGY*}

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* {units=*NX_LENGTH*}

Distance of the point of evaluation of the beam from the sample surface.

incident_energy: (required) *NX_FLOAT* {units=*NX_ENERGY*} <=

incident_energy_spread: (recommended) *NX_NUMBER* {units=*NX_ENERGY*}

incident_polarization: (recommended) *NX_NUMBER* {units=*NX_ANY*} <=

ELECTRONANALYSER: (required) *NXelectronanalyser*

description: (required) *NX_CHAR* <=

energy_resolution: (recommended) *NX_FLOAT* {units=*NX_ENERGY*} <=

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* {units=*NX_ENERGY*} <=

energy_scan_mode: (required) *NX_CHAR* <=

entrance_slit: (optional) *NXaperture* <=

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) *NXaperture* <=

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* {units=*NX_TEMPERATURE*} *<=*

drain_current: (recommended) *NX_FLOAT* {units=*NX_CURRENT*} *<=*

sample_bias: (recommended) *NX_FLOAT* {units=*NX_CURRENT*} *<=*

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.

atom_types: (recommended) *NX_CHAR*

List of comma-separated elements from the periodic table that are contained in the sample. If the sample substance has multiple components, all elements from each component must be included in *atom_types*.

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* {units=*NX_PRESSURE*}

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* {units=*NX_ANY*} *<=*

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/slow_axes-field*
- */NXmpes/ENTRY/INSTRUMENT/energy_resolution-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR-group*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/drain_current-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_bias-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_temperature-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/atom_types-field*
- */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*
- */NXmpes/ENTRY/USER-group*

- */NXmpes/ENTRY/USER/address-field*
- */NXmpes/ENTRY/USER/affiliation-field*
- */NXmpes/ENTRY/USER/email-field*
- */NXmpes/ENTRY/USER/name-field*
- */NXmpes/ENTRY/USER/orcid-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmpes.nxdl.xml

NXms

Status:

application definition, extends *NXobject*

Description:

Application definition, spatiotemporal characterization of a microstructure.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_b: Number of boundaries of the bounding box or primitive to domain.

n_p: Number of parameter required for the chosen orientation parameterization.

c: Number of texture components identified.

Groups cited:
NXcg_grid, *NXcg_point_set*, *NXcg_polyhedron_set*, *NXcg_roi_set*, *NXcoordinate_system_set*, *NXdata*, *NXem_ebsd_conventions*, *NXentry*, *NXms_feature_set*, *NXms_snapshot_set*, *NXms_snapshot*, *NXorientation_set*, *NXprocess*, *NXprogram*, *NXsample*, *NXtransformations*, *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: `NXms`

workflow_identifier: (required) *NX_CHAR*

Ideally, a (globally) unique persistent identifier for referring to this experiment or computer simulation.

The identifier is usually defined/issued by the facility, laboratory, or the principle investigator. The identifier enables to link experiments to e.g. proposals.

workflow_description: (optional) *NX_CHAR*

Free-text description about the workflow (experiment/analysis/simulation).

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 characterization started.

end_time: (recommended) *NX_DATE_TIME <=*

ISO 8601 time code with local time zone offset to UTC included when the characterization ended.

experiment_or_simulation: (required) *NX_CHAR*

Specify if the (characterization) results/data of this instance of an application definition are based on the results of a simulation or the results of a post-processing of measured data to describe a microstructure.

The term microstructure is used to describe the spatial arrangement of crystal defects inside a sample/specimen without demanding necessarily that this structure is mainly at the micron length scale. Nanostructure and macrostructure are close synonyms. Material architecture is a narrow synonym.

Given that microstructure simulations nowadays more and more consider the atomic arrangement, this application definition can also be used to describe features at the scale of atoms.

Any of these values: `experiment` | `simulation`

PROGRAM: (required) *NXprogram*

program_name: (required) *NX_CHAR*

@version: (required) *NX_CHAR*

USER: (optional) *NXuser <=*

Contact information and eventually details of at least one person involved in creating this result. 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: (recommended) *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. If this field is field the specific service should also be written in orcid_platform

orcid_platform: (recommended) *NX_CHAR <=*

Name of the OrcID or ResearcherID where the account under orcid is registered.

telephone_number: (optional) *NX_CHAR <=*

(Business) (tele)phone number of the user at the point in time when the experiment was performed.

role: (recommended) *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.

DATA: (optional) *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. A least a right-handed Cartesian coordinate system with base vectors named x, y, and z has to be specified. Each base vector of the coordinate system should be described with an NXtransformations instance. **TRANSFORMATIONS**: (`minOccurs=3`) *NXtransformations <=*

conventions: (required) *NXem_ebsd_conventions*

rotation_conventions: (required) *NXprocess <=*

three_dimensional_rotation_handedness: (required) *NX_CHAR <=*

rotation_convention: (required) *NX_CHAR <=*

euler_angle_convention: (required) *NX_CHAR <=*

axis_angle_convention: (required) *NX_CHAR <=*

orientation_parameterization_sign_convention: (required) *NX_CHAR <=*

processing_reference_frame: (required) *NXprocess <=*

reference_frame_type: (required) *NX_CHAR <=*

xaxis_direction: (required) *NX_CHAR <=*

xaxis_alias: (required) *NX_CHAR <=*

yaxis_direction: (required) *NX_CHAR <=*

yaxis_alias: (required) *NX_CHAR* <=

zaxis_direction: (required) *NX_CHAR* <=

zaxis_alias: (required) *NX_CHAR* <=

origin: (required) *NX_CHAR* <=

sample_reference_frame: (required) *NXprocess* <=

reference_frame_type: (required) *NX_CHAR* <=

xaxis_direction: (required) *NX_CHAR* <=

yaxis_direction: (required) *NX_CHAR* <=

zaxis_direction: (required) *NX_CHAR* <=

origin: (required) *NX_CHAR* <=

ROI_SET: (required) *NXcg_roi_set*

The simulated or characterized material volume element aka domain. At least one instance of geometry required either NXcg_grid, NXcg_polyhedron_set, or NXcg_point_set. This geometry group needs to contain details about the boundary conditions. **grid**: (optional) *NXcg_grid*

point_set: (optional) *NXcg_point_set*

polyhedron_set: (optional) *NXcg_polyhedron_set*

boundary: (optional) *NXcg_polyhedron_set*

A boundary to the volume element. Either an instance of NXcg_hexahedron_set or of NXcg_ellipsoid_set. **number_of_boundaries**: (required) *NX_POSINT* {units=*NX_UNITLESS*}

How many distinct boundaries are distinguished. Value required equal to n_b.

boundaries: (required) *NX_CHAR* (Rank: 1, Dimensions: [n_b])

Name of the boundaries. E.g. left, right, front, back, bottom, top,

boundary_conditions: (required) *NX_UINT* (Rank: 1, Dimensions: [n_b]) {units=*NX_UNITLESS*}

The boundary conditions for each boundary:

0 - undefined 1 - open 2 - periodic 3 - mirror 4 - von Neumann 5 - Dirichlet

snapshot_set: (required) *NXms_snapshot_set*

Collection of microstructural data observed/simulated.

identifier_offset: (required) *NX_UINT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing snapshots. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

evolution: (optional) *NXprocess*

Summary quantities which are the result of some post-processing of the snapshot data (averaging, integrating, interpolating). Frequently used descriptors from continuum mechanics and thermodynamics can be used here. A few examples are given. Each descriptor is currently modelled as an instance of an NXprocess because it is relevant to understand how the descriptors are computed. **temperature**: (optional) *NXprocess*

pressure: (optional) *NXprocess*

stress: (optional) *NXprocess*

strain: (optional) *NXprocess*

deformation_gradient: (optional) *NXprocess*

magnetic_field: (optional) *NXprocess*

electric_field: (optional) *NXprocess*

MS_SNAPSHOT: (required) *NXms_snapshot* <=

time: (required) *NX_NUMBER* {units=*NX_TIME*} <=

Measured or simulated physical time stamp for this snapshot. Not to be confused with wall-clock timing or profiling data.

iteration: (required) *NX_UINT* {units=*NX_UNITLESS*}

Iteration or increment counter.

grid: (optional) *NXcg_grid*

polyhedron_set: (optional) *NXcg_polyhedron_set*

point_set: (optional) *NXcg_point_set*

MS_FEATURE_SET: (optional) *NXms_feature_set*

Conceptually distinguished object/feature in the ROI/ system with some relevance. Instances of NXms_feature_set can be nested to build a hierarchy of logically-related objects.

A typical example for MD simulations is to have one ms_feature_set for the atoms which is the parent to another ms_feature_set for monomers/molecules/proteins which is then the parent to another ms_feature_set for the secondary, another feature_set for the tertiary, and the parent for another feature_set for the quaternary structure.

Another typical example from materials engineering is to have one ms_feature_set for crystals (grains/phases) which serves as the parent to another ms_feature_set for interfaces between these crystals which then is the parent for another ms_feature_set to describe the triple junctions which is then the parent for the quadruple/higher-order junctions between which connect the triple lines.

Another example from discrete dislocation dynamics could be to have one ms_feature_set for the dislocations (maybe separate sets for each dislocation type or family) which are then parents to other ms_feature_set which describe junctions between dislocations or features along the dislocation line network.

odf: (optional) *NXprocess*

Details about the orientation distribution function within the entire domain. **computation_method**: (required) *NX_CHAR*

With which method was the ODF approximated?

texture_index: (optional) *NX_NUMBER* {units=*NX_ANY*}

TO BE DEFINED

texture_strength: (optional) *NX_NUMBER* {units=*NX_ANY*}

TO BE DEFINED

volume_statistics: (optional) *NXorientation_set*

Collection of texture components commonly referred to.

parameterization: (required) *NX_CHAR* *<=*

Any of these values: `bunge-euler (ZXZ)` | `quaternion`

orientation: (required) *NX_NUMBER* (Rank: 2, Dimensions: [c, n_p]) {units=*NX_ANY*} *<=*

Parameterized orientations.

name: (required) *NX_CHAR* (Rank: 1, Dimensions: [c])

Name of each texture component, e.g. Cube, Dillamore, Y.

integration_radius: (required) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_ANGLE*}

The portion of orientation space integrated over to compute the volume fraction.

volume_fraction: (required) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_DIMENSIONLESS*}

The volume fraction of each texture component.

TRANSFORMATIONS: (required) *NXtransformations* *<=*

Reference to or definition of a coordinate system with which the definitions are interpretable.

modf: (optional) *NXprocess*

Details about the disorientation distribution function within the entire domain.

gbcd: (optional) *NXprocess*

Details about the grain boundary character distribution within the entire domain.

temperature: (optional) *NXprocess*

pressure: (optional) *NXprocess*

stress: (optional) *NXprocess*

strain: (optional) *NXprocess*

deformation_gradient: (optional) *NXprocess*

magnetic_field: (optional) *NXprocess*

electric_field: (optional) *NXprocess*

recrystallization_kinetics: (optional) *NXprocess*

grain_size_distribution: (optional) *NXprocess*

recrystallization_front: (optional) *NXprocess*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXms/ENTRY-group*
- */NXms/ENTRY/conventions-group*
- */NXms/ENTRY/conventions/processing_reference_frame-group*
- */NXms/ENTRY/conventions/processing_reference_frame/origin-field*
- */NXms/ENTRY/conventions/processing_reference_frame/reference_frame_type-field*
- */NXms/ENTRY/conventions/processing_reference_frame/xaxis_alias-field*
- */NXms/ENTRY/conventions/processing_reference_frame/xaxis_direction-field*
- */NXms/ENTRY/conventions/processing_reference_frame/yaxis_alias-field*
- */NXms/ENTRY/conventions/processing_reference_frame/yaxis_direction-field*
- */NXms/ENTRY/conventions/processing_reference_frame/zaxis_alias-field*
- */NXms/ENTRY/conventions/processing_reference_frame/zaxis_direction-field*
- */NXms/ENTRY/conventions/rotation_conventions-group*
- */NXms/ENTRY/conventions/rotation_conventions/axis_angle_convention-field*
- */NXms/ENTRY/conventions/rotation_conventions/euler_angle_convention-field*
- */NXms/ENTRY/conventions/rotation_conventions/orientation_parameterization_sign_convention-field*
- */NXms/ENTRY/conventions/rotation_conventions/rotation_convention-field*
- */NXms/ENTRY/conventions/rotation_conventions/three_dimensional_rotation_handedness-field*
- */NXms/ENTRY/conventions/sample_reference_frame-group*
- */NXms/ENTRY/conventions/sample_reference_frame/origin-field*
- */NXms/ENTRY/conventions/sample_reference_frame/reference_frame_type-field*
- */NXms/ENTRY/conventions/sample_reference_frame/xaxis_direction-field*
- */NXms/ENTRY/conventions/sample_reference_frame/yaxis_direction-field*
- */NXms/ENTRY/conventions/sample_reference_frame/zaxis_direction-field*
- */NXms/ENTRY/COORDINATE_SYSTEM_SET-group*
- */NXms/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-group*
- */NXms/ENTRY/DATA-group*
- */NXms/ENTRY/definition-field*

- */NXms/ENTRY/end_time-field*
- */NXms/ENTRY/experiment_or_simulation-field*
- */NXms/ENTRY/PROGRAM-group*
- */NXms/ENTRY/PROGRAM/program_name-field*
- */NXms/ENTRY/PROGRAM/program_name@version-attribute*
- */NXms/ENTRY/ROI_SET-group*
- */NXms/ENTRY/ROI_SET/boundary-group*
- */NXms/ENTRY/ROI_SET/boundary/boundaries-field*
- */NXms/ENTRY/ROI_SET/boundary/boundary_conditions-field*
- */NXms/ENTRY/ROI_SET/boundary/number_of_boundaries-field*
- */NXms/ENTRY/ROI_SET/grid-group*
- */NXms/ENTRY/ROI_SET/point_set-group*
- */NXms/ENTRY/ROI_SET/polyhedron_set-group*
- */NXms/ENTRY/ROI_SET/snapshot_set-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/evolution-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/evolution/deformation_gradient-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/evolution/electric_field-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/evolution/magnetic_field-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/evolution/pressure-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/evolution/strain-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/evolution/stress-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/evolution/temperature-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/identifier_offset-field*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/deformation_gradient-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/electric_field-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/gbcd-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/grain_size_distribution-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/grid-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/iteration-field*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/magnetic_field-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/modf-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/MS_FEATURE_SET-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/odf-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/odf/computation_method-field*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/odf/texture_index-field*

- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/odf/texture_strength-field*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/odf/volume_statistics-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/odf/volume_statistics/integration_radius-field*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/odf/volume_statistics/name-field*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/odf/volume_statistics/orientation-field*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/odf/volume_statistics/parameterization-field*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/odf/volume_statistics/TRANSFORMATIONS-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/odf/volume_statistics/volume_fraction-field*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/point_set-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/polyhedron_set-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/pressure-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/recrystallization_front-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/recrystallization_kinetics-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/strain-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/stress-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/temperature-group*
- */NXms/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/time-field*
- */NXms/ENTRY/specimen-group*
- */NXms/ENTRY/specimen/name-field*
- */NXms/ENTRY/start_time-field*
- */NXms/ENTRY/USER-group*
- */NXms/ENTRY/USER/address-field*
- */NXms/ENTRY/USER/affiliation-field*
- */NXms/ENTRY/USER/email-field*
- */NXms/ENTRY/USER/name-field*
- */NXms/ENTRY/USER/orcid-field*
- */NXms/ENTRY/USER/orcid_platform-field*
- */NXms/ENTRY/USER/role-field*
- */NXms/ENTRY/USER/social_media_name-field*
- */NXms/ENTRY/USER/social_media_platform-field*
- */NXms/ENTRY/USER/telephone_number-field*
- */NXms/ENTRY/workflow_description-field*
- */NXms/ENTRY/workflow_identifier-field*
- */NXms/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXms.nxdl.xml

NXms_feature_set

Status:

base class, extends *NXobject*

Description:

Set of topological/spatial features in materials build from other features.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: Dimensionality

c: Cardinality/number of members/features in the feature set.

n_type_dict: Number of types in the dictionary of human-readable types of features.

n_parent_ids: Total number of parent identifier.

Groups cited:
NXcg_point_set, *NXcg_polyhedron_set*, *NXcg_polyline_set*, *NXcg_triangle_set*, *NXprocess*

Structure:

@depends_on: (optional) *NX_CHAR*

Name (or link?) to another NXms_feature_set which defines features what are assumed as the parents/super_features of all members in this feature set. If depends_on is set to "." or this attribute is not defined in an instance of this application definition, assume that this feature_set instance represents the root feature_set of the feature tree/graph.

dimensionality: (optional) *NX_UINT*

What is the best matching description how dimensional the feature is.

Any of these values: `0` | `1` | `2` | `3`

cardinality: (optional) *NX_UINT* {units=*NX_UNITLESS*}

How many features/members are in this set?

type_dict_keyword: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_type_dict])

The keywords of the dictionary of human-readable types of features. Using terms from a community-agreed upon controlled vocabulary, e.g. atom, solute, vacancy, monomer, molecule, anti-site, crowd_ion, quadruple junction, triple line, disconnection, dislocation, kink, jog, stacking_fault, homo_phase_boundary, hetero_phase_boundary, surface, thermal_groove_root, precipitate, dispersoid, pore, crack is recommended.

type_dict_value: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_type_dict]) {units=*NX_UNITLESS*}

The integer identifier used to resolve of which type each feature is, i.e. the values of the dictionary of human-readable types of features.

number_of_parent_identifier: (optional) *NX_UINT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

For each feature in the set specify the associated number of identifier to parent features as are resolvable via depends_on. This array enables to compute the array interval from which details for specific features can be extracted as if they would be stored in an own group.

parent_identifier: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_parent_ids]) {units=*NX_UNITLESS*}

Concatenated array of parent identifier for each feature (in the sequence) according to identifier and how the identifier of features in the here described feature set are defined (implicitly from 0, to c-1 or via explicit identifier.

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing features. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Integer used to distinguish features for explicit indexing.

points: (optional) *NXprocess*

Assumptions and parameter to arrive at geometric primitives to locate zero-dimensional/point-(like) features which are discretized/represented by points. **geometry**: (optional) *NXcg_point_set*

uncertainty: (optional) *NXprocess*

Assumptions, parameters, and details how positional uncertainty of the geometry is quantified.

lines: (optional) *NXprocess*

Assumptions and parameters to arrive at geometric primitives to locate one-dimensional/line-like features which are discretized by polylines. **geometry**: (optional) *NXcg_polyline_set*

uncertainty: (optional) *NXprocess*

Assumptions, parameters, and details how positional uncertainty of the geometry is quantified.

interfaces: (optional) *NXprocess*

Assumptions and parameters to arrive at geometric primitives to locate two-dimensional/interface features which are discretized by triangulated surface meshes. **geometry**: (optional) *NXcg_triangle_set*

uncertainty: (optional) *NXprocess*

Assumptions, parameters, and details how positional uncertainty of the geometry is quantified.

volumes: (optional) *NXprocess*

Assumptions and parameters to arrive at geometric primitives to locate three-dimensional/volumetric features which are discretized by triangulated surface meshes of polyhedra. **geometry**: (optional) *NXcg_polyhedron_set*

uncertainty: (optional) *NXprocess*

Assumptions, parameters, and details how positional uncertainty of the geometry is quantified.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXms_feature_set/cardinality-field*
- */NXms_feature_set/dimensionality-field*
- */NXms_feature_set/identifier-field*
- */NXms_feature_set/identifier_offset-field*
- */NXms_feature_set/interfaces-group*
- */NXms_feature_set/interfaces/geometry-group*
- */NXms_feature_set/interfaces/uncertainty-group*
- */NXms_feature_set/lines-group*
- */NXms_feature_set/lines/geometry-group*
- */NXms_feature_set/lines/uncertainty-group*
- */NXms_feature_set/number_of_parent_identifier-field*
- */NXms_feature_set/parent_identifier-field*
- */NXms_feature_set/points-group*
- */NXms_feature_set/points/geometry-group*
- */NXms_feature_set/points/uncertainty-group*
- */NXms_feature_set/type_dict_keyword-field*
- */NXms_feature_set/type_dict_value-field*
- */NXms_feature_set/volumes-group*
- */NXms_feature_set/volumes/geometry-group*
- */NXms_feature_set/volumes/uncertainty-group*
- */NXms_feature_set@depends_on-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXms_feature_set.nxdl.xml

NXms_score_config

Status:

application definition, extends *NXobject*

Description:

Application definition to control a simulation with the SCORE model.

Symbols:

n_dg_ori: Number of Bunge-Euler angle triplets for deformed grains.

n_rx_ori: Number of Bunge-Euler angle triplets for recrystallization nuclei.

n_su: Number of solitary unit domains to export.

Groups cited:
NXcollection, *NXcs_profiling*, *NXentry*, *NXprocess*, *NXprogram*

Structure:

ENTRY: (required) *NXentry*

@version: (required) *NX_CHAR*

Version specifier of this application definition.

definition: (required) *NX_CHAR <=*

Official NeXus NXDL schema with which this file was written.

Obligatory value: `NXms_score_config`

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally persistent) unique identifier for referring to this analysis.

analysis_description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

time_stamp: (required) *NX_DATE_TIME*

ISO 8601 formatted time code with local time zone offset to UTC information included when this configuration file was created.

PROGRAM: (required) *NXprogram*

program_name: (required) *NX_CHAR*

@version: (required) *NX_CHAR*

initial_microstructure: (required) *NXprocess <=*

Relevant data to instantiate a starting configuration that is typically a microstructure in deformed conditions where stored (elastic) energy is stored in the form of crystal defects, which in the SCORE model are is considered as mean-field dislocation content.

type: (required) *NX_CHAR*

Which model should be used to generate a starting microstructure.

Any of these values: `cuboidal` | `poisson_voronoi` | `ebsd` | `damask`

cell_size: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Edge length of the cubic cells of each CA domain.

domain_size: (required) *NX_UINT* (Rank: 1, Dimensions: [3]) {units=*NX_UNITLESS*}

Extend of each CA domain in voxel along the x, y, and z direction. Deformation of sheet material is assumed. The x axis is assumed pointing along the rolling direction. The y axis is assumed pointing along the transverse direction. The z axis is assumed pointing along the normal direction.

grain_size: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

Extent of each deformed grain along the x, y, and z direction when type is cuboidal.

grain_diameter: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Average spherical diameter when type is poisson_voronoi.

grain_euler: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_dg_ori, 3]) {units=*NX_ANGLE*}

Set of Bunge-Euler angles to sample orientations randomly from.

ebsd: (optional) *NXprocess*

The EBSD dataset from which the initial microstructure is instantiated if initial_microstructure/type has value ebsd. **filename**: (required) *NX_CHAR*

Path and name of the EBSD dataset from which to generate the starting microstructure. **@version**: (required) *NX_CHAR*

SHA256 checksum of the file which contains the EBSD dataset.

stepsize: (required) *NX_FLOAT* (Rank: 1, Dimensions: [2]) {units=*NX_LENGTH*}

Size of the EBSD. The EBSD orientation mapping has to be on a regular grid of square-shaped pixels.

nucleation_model: (required) *NXprocess <=*

Phenomenological model according to which it nuclei are placed into the domain and assumed growing. **spatial_distribution_model**: (required) *NX_CHAR*

According to which model will the nuclei become distributed spatially. CSR means complete spatial randomness. Custom is implementation specific. GB places nuclei at grain boundaries.

Any of these values: `csr` | `custom` | `gb`

incubation_time_model: (required) *NX_CHAR*

According to which model will the nuclei start to grow.

Obligatory value: `site_saturation`

orientation_model: (required) *NX_CHAR*

According to which model will the nuclei get their orientation assigned.

Obligatory value: `sample_from_nucleus_euler`

nucleus_euler: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_rx_ori, 3]) {units=*NX_ANGLE*}

Set of Bunge-Euler angles to sample orientations of nuclei randomly from.

material_properties: (required) *NXprocess <=*

(Mechanical) properties of the material which scale the amount of stored (elastic) energy in the ROI/system. **reference_shear_modulus**: (required) *NX_FLOAT* {units=*NX_PRESSURE*}

Shear modulus at zero Kelvin.

reference_burgers_magnitude: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Magnitude at the Burgers vector at zero Kelvin.

melting_temperature: (required) *NX_FLOAT* {units=*NX_TEMPERATURE*}

Melting temperature in degrees Celsius.

grain_boundary_mobility_model: (required) *NXprocess <=*

Model for the assumed mobility of grain boundaries with different disorientation. **model**: (required) *NX_CHAR*

Which type of fundamental model for the grain boundary mobility: For the Sebald-Gottstein model the following equation is used. For the Rollett-Holm model the following equation is used.

Any of these values: `sebald_gottstein` | `rollett_holm`

sebald_gottstein_parameters: (required) *NXcollection*

lagb_pre_factor: (required) *NX_FLOAT* {units=*NX_ANY*}

lagb_enthalpy: (required) *NX_FLOAT* {units=*NX_ANY*}

hagb_pre_factor: (required) *NX_FLOAT* {units=*NX_ANY*}

hagb_enthalpy: (required) *NX_FLOAT* {units=*NX_ANY*}

special_pre_factor: (required) *NX_FLOAT* {units=*NX_ANY*}

special_enthalpy: (required) *NX_FLOAT* {units=*NX_ANY*}

rollett_holm_parameters: (required) *NXcollection*

hagb_pre_factor: (required) *NX_FLOAT* {units=*NX_ANY*}

hagb_enthalpy: (required) *NX_FLOAT* {units=*NX_ANY*}

lagb_to_hagb_cut: (required) *NX_FLOAT* {units=*NX_ANY*}

lagb_to_hagb_transition: (required) *NX_FLOAT* {units=*NX_ANY*}

lagb_to_hagb_exponent: (required) *NX_FLOAT* {units=*NX_ANY*}

stored_energy_recovery_model: (required) *NXprocess <=*

Simulated evolution of the dislocation density as the stored (elastic) energy assumed stored in each grain. **model**: (required) *NX_CHAR*

Which type of recovery model.

Obligatory value: `none`

dispersoid_drag_model: (required) *NXprocess <=*

Simulated reduction of the grain boundary speed due to the presence of dispersoids through which the total grain boundary area of the recrystallization front can be reduced. **model**: (required) *NX_CHAR*

Which type of drag model.

Any of these values: `none` | `zener_smith`

zener_smith_parameter: (required) *NXcollection*

pre_factor: (required) *NX_FLOAT*

surface_energy_density: (required) *NX_FLOAT*

time: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_TIME*}

Support point of the linearized curve of simulated time matching a specific support point of the average dispersoid radius.

radius: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_LENGTH*}

Support point of the linearized curve of the average dispersoid radius.

time_temperature_history: (required) *NXprocess* <=

Simulated time temperature profile

time: (required) *NX_FLOAT* (Rank: 1, Dimensions: [j]) {units=*NX_TIME*}

Support point of the linearized curve of simulated time matching a specific support point of the temperature.

temperature: (required) *NX_FLOAT* (Rank: 1, Dimensions: [j]) {units=*NX_LENGTH*}

Support point of the linearized curve of the temperature.

stop_criteria: (required) *NXprocess* <=

Criteria which enable to stop the simulation in a controlled manner. Whichever criterion is fulfilled first stops the simulation. **max_x**: (required) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Maximum recrystallized volume fraction.

max_time: (required) *NX_NUMBER* {units=*NX_TIME*}

Maximum simulated physical time.

max_iteration: (required) *NX_UINT* {units=*NX_UNITLESS*}

Maximum number of iteration steps.

numerics: (required) *NXprocess* <=

Settings relevant for stable numerical integration.

max_delta_x: (required) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Maximum fraction equivalent to the migration of the fastest grain boundary in the system how much a cell may be consumed in a single iteration.

initial_cell_cache: (required) *NX_FLOAT* {units=*NX_UNITLESS*}

Fraction of the total number of cells in the CA which should initially be allocated for offering cells in the recrystallization front.

realloc_cell_cache: (required) *NX_FLOAT* {units=*NX_UNITLESS*}

By how much more times should the already allocated memory be is increased to offer space for storing states of cells in the recrystallization front.

defragment_cell_cache: (required) *NX_BOOLEAN*

Should the cache for cells in the recrystallization front be defragmented on-the-fly.

defragment_x: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_DIMENSIONLESS*}

Heuristic recrystallized volume target values at which the cache for cells in the recrystallization front will be defragmented on-the-fly.

snapshot_x: (required) *NX_FLOAT* (Rank: 1, Dimensions: [j]) {units=*NX_DIMENSIONLESS*}

List of recrystallized volume target values at which a snapshot of the CA state should be exported for.

solitary_unit_model: (required) *NXprocess* <=

apply: (required) *NX_BOOLEAN*

Perform a statistical analyses of the results as it was proposed by M. Kühbach (solitary unit model ensemble approach).

number_of_domains: (required) *NX_UINT* {units=*NX_UNITLESS*}

How many independent cellular automaton domains should be instantiated.

rediscretization: (required) *NX_UINT* {units=*NX_UNITLESS*}

Into how many time steps should the real time interval be discretized upon during post-processing the results with the solitary unit modeling approach.

domain_identifier: (required) *NX_UINT* (Rank: 1, Dimensions: [n_su]) {units=*NX_UNITLESS*}

List of identifier for those domain which should be rendered. Identifier start from 0.

performance: (required) *NXcs_profiling*

current_working_directory: (required) *NX_CHAR* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXms_score_config/ENTRY-group*
- */NXms_score_config/ENTRY/analysis_description-field*
- */NXms_score_config/ENTRY/analysis_identifier-field*
- */NXms_score_config/ENTRY/definition-field*
- */NXms_score_config/ENTRY/dispersoid_drag_model-group*
- */NXms_score_config/ENTRY/dispersoid_drag_model/model-field*
- */NXms_score_config/ENTRY/dispersoid_drag_model/zener_smith_parameter-group*
- */NXms_score_config/ENTRY/dispersoid_drag_model/zener_smith_parameter/pre_factor-field*
- */NXms_score_config/ENTRY/dispersoid_drag_model/zener_smith_parameter/radius-field*
- */NXms_score_config/ENTRY/dispersoid_drag_model/zener_smith_parameter/surface_energy_density-field*
- */NXms_score_config/ENTRY/dispersoid_drag_model/zener_smith_parameter/time-field*
- */NXms_score_config/ENTRY/grain_boundary_mobility_model-group*
- */NXms_score_config/ENTRY/grain_boundary_mobility_model/model-field*
- */NXms_score_config/ENTRY/grain_boundary_mobility_model/rollett_holm_parameters-group*
- */NXms_score_config/ENTRY/grain_boundary_mobility_model/rollett_holm_parameters/hagb_enthalpy-field*
- */NXms_score_config/ENTRY/grain_boundary_mobility_model/rollett_holm_parameters/hagb_pre_factor-field*

- */NXms_score_config/ENTRY/grain_boundary_mobility_model/rollett_holm_parameters/lagb_to_hagb_cut-field*
- */NXms_score_config/ENTRY/grain_boundary_mobility_model/rollett_holm_parameters/lagb_to_hagb_exponent-field*
- */NXms_score_config/ENTRY/grain_boundary_mobility_model/rollett_holm_parameters/lagb_to_hagb_transition-field*
- */NXms_score_config/ENTRY/grain_boundary_mobility_model/sebald_gottstein_parameters-group*
- */NXms_score_config/ENTRY/grain_boundary_mobility_model/sebald_gottstein_parameters/hagb_enthalpy-field*
- */NXms_score_config/ENTRY/grain_boundary_mobility_model/sebald_gottstein_parameters/hagb_pre_factor-field*
- */NXms_score_config/ENTRY/grain_boundary_mobility_model/sebald_gottstein_parameters/lagb_enthalpy-field*
- */NXms_score_config/ENTRY/grain_boundary_mobility_model/sebald_gottstein_parameters/lagb_pre_factor-field*
- */NXms_score_config/ENTRY/grain_boundary_mobility_model/sebald_gottstein_parameters/special_enthalpy-field*
- */NXms_score_config/ENTRY/grain_boundary_mobility_model/sebald_gottstein_parameters/special_pre_factor-field*
- */NXms_score_config/ENTRY/initial_microstructure-group*
- */NXms_score_config/ENTRY/initial_microstructure/cell_size-field*
- */NXms_score_config/ENTRY/initial_microstructure/domain_size-field*
- */NXms_score_config/ENTRY/initial_microstructure/ebsd-group*
- */NXms_score_config/ENTRY/initial_microstructure/ebsd/filename-field*
- */NXms_score_config/ENTRY/initial_microstructure/ebsd/filename@version-attribute*
- */NXms_score_config/ENTRY/initial_microstructure/ebsd/stepsize-field*
- */NXms_score_config/ENTRY/initial_microstructure/grain_diameter-field*
- */NXms_score_config/ENTRY/initial_microstructure/grain_euler-field*
- */NXms_score_config/ENTRY/initial_microstructure/grain_size-field*
- */NXms_score_config/ENTRY/initial_microstructure/type-field*
- */NXms_score_config/ENTRY/material_properties-group*
- */NXms_score_config/ENTRY/material_properties/melting_temperature-field*
- */NXms_score_config/ENTRY/material_properties/reference_burgers_magnitude-field*
- */NXms_score_config/ENTRY/material_properties/reference_shear_modulus-field*
- */NXms_score_config/ENTRY/nucleation_model-group*
- */NXms_score_config/ENTRY/nucleation_model/incubation_time_model-field*
- */NXms_score_config/ENTRY/nucleation_model/nucleus_euler-field*
- */NXms_score_config/ENTRY/nucleation_model/orientation_model-field*
- */NXms_score_config/ENTRY/nucleation_model/spatial_distribution_model-field*

- */NXms_score_config/ENTRY/numerics-group*
- */NXms_score_config/ENTRY/numerics/defragment_cell_cache-field*
- */NXms_score_config/ENTRY/numerics/defragment_x-field*
- */NXms_score_config/ENTRY/numerics/initial_cell_cache-field*
- */NXms_score_config/ENTRY/numerics/max_delta_x-field*
- */NXms_score_config/ENTRY/numerics/realloc_cell_cache-field*
- */NXms_score_config/ENTRY/numerics/snapshot_x-field*
- */NXms_score_config/ENTRY/performance-group*
- */NXms_score_config/ENTRY/performance/current_working_directory-field*
- */NXms_score_config/ENTRY/PROGRAM-group*
- */NXms_score_config/ENTRY/PROGRAM/program_name-field*
- */NXms_score_config/ENTRY/PROGRAM/program_name@version-attribute*
- */NXms_score_config/ENTRY/results_path-field*
- */NXms_score_config/ENTRY/solitary_unit_model-group*
- */NXms_score_config/ENTRY/solitary_unit_model/apply-field*
- */NXms_score_config/ENTRY/solitary_unit_model/domain_identifier-field*
- */NXms_score_config/ENTRY/solitary_unit_model/number_of_domains-field*
- */NXms_score_config/ENTRY/solitary_unit_model/rediscretization-field*
- */NXms_score_config/ENTRY/stop_criteria-group*
- */NXms_score_config/ENTRY/stop_criteria/max_iteration-field*
- */NXms_score_config/ENTRY/stop_criteria/max_time-field*
- */NXms_score_config/ENTRY/stop_criteria/max_x-field*
- */NXms_score_config/ENTRY/stored_energy_recovery_model-group*
- */NXms_score_config/ENTRY/stored_energy_recovery_model/model-field*
- */NXms_score_config/ENTRY/time_stamp-field*
- */NXms_score_config/ENTRY/time_temperature_history-group*
- */NXms_score_config/ENTRY/time_temperature_history/temperature-field*
- */NXms_score_config/ENTRY/time_temperature_history/time-field*
- */NXms_score_config/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXms_score_config.nxdl.xml

NXms_score_results

Status:

application definition, extends *NXobject*

Description:

Application definition for storing results of the SCORE cellular automaton.

The SCORE cellular automaton model for primary recrystallization is an example of typical materials engineering applications used within the field of so-called Integral Computational Materials Engineering (ICME) whereby one can simulate the evolution of microstructures.

Specifically the SCORE model can be used to simulate the growth of static recrystallization nuclei. The model is described in the literature:

- M. Kühbach et al.
- C. Haase et al.
- M. Diehl et al.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_b: Number of boundaries of the bounding box or primitive to domain.

n_p: Number of parameter required for chosen orientation parameterization.

n_tex: Number of texture components identified.

d: Dimensionality.

c: Cardinality.

n_front: Number of active cells in the (recrystallization) front.

n_grains: Number of grains in the computer simulation.

Groups cited:

NXcg_grid, *NXcg_polyhedron_set*, *NXcg_roi_set*, *NXcoordinate_system_set*, *NXcs_computer*, *NXcs_cpu*, *NXcs_gpu*, *NXcs_io_obj*, *NXcs_io_sys*, *NXcs_mm_sys*, *NXcs_profiling_event*, *NXcs_profiling*, *NXdata*, *NXem_ebsd_conventions*, *NXentry*, *NXfabrication*, *NXms_snapshot_set*, *NXms_snapshot*, *NXprocess*, *NXprogram*, *NXsample*, *NXtransformations*, *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: `NXms_score_results`

analysis_identifier: (required) *NX_CHAR*

Ideally, a (globally) unique persistent identifier for referring to this computer simulation.

The identifier is usually defined/issued by the facility, laboratory, or the principle investigator. The identifier enables to link experiments to e.g. proposals.

analysis_description: (optional) *NX_CHAR*

Free-text description about the workflow (analysis/simulation).

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 characterization started.

end_time: (required) *NX_DATE_TIME* <=

ISO 8601 time code with local time zone offset to UTC included when the characterization ended.

experiment_or_simulation: (required) *NX_CHAR*

Specify if the (characterization) results/data of this instance of an application definition are based on the results of a simulation or the results of a post-processing of measured data to describe a microstructure. The term microstructure is used to describe the spatial arrangement of crystal defects inside a sample/specimen without demanding necessarily that this structure is mainly at the micron length scale. Nanostructure and macrostructure are close synonyms. Material architecture is a narrow synonym.

Any of these values: `experiment` | `simulation`

config_filename: (required) *NX_CHAR*

The path and name of the config file for this analysis.

@version: (required) *NX_CHAR*

At least SHA256 strong hash of the specific config_file for tracking provenance.

results_path: (optional) *NX_CHAR*

Path to the directory where the tool should store NeXus/HDF5 results of this analysis. If not specified results will be stored in the current working directory.

status: (required) *NX_CHAR*

A statement whether the SCORE executable managed to process the analysis or failed prematurely.

This status is written to the results file after the end_time at which point the executable must not compute any analysis. Only when this status message is present and shows *success*, the user should consider the results. In all other cases it might be that the executable has terminated prematurely or another error occurred.

Any of these values: `success` | `failure`

PROGRAM: (required) *NXprogram*

program_name: (required) *NX_CHAR*

@version: (required) *NX_CHAR*

USER: (optional) *NXuser* <=

Contact information and eventually details of at least one person involved in creating this result. 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: (recommended) *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. If this field is field the specific service should also be written in orcid_platform

orcid_platform: (recommended) *NX_CHAR* <=

Name of the OrcID or ResearcherID where the account under orcid is registered.

telephone_number: (optional) *NX_CHAR* <=

(Business) (tele)phone number of the user at the point in time when the experiment was performed.

role: (recommended) *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.

DATA: (optional) *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. A least a right-handed Cartesian coordinate system with base vectors named x, y, and z has to be specified.

Each base vector of the coordinate system should be described with an NXtransformations instance. **TRANSFORMATIONS**: (minOccurs=3) *NXtransformations <=*

conventions: (required) *NXem_ebsd_conventions*

rotation_conventions: (required) *NXprocess <=*

three_dimensional_rotation_handedness: (required) *NX_CHAR <=*

rotation_convention: (required) *NX_CHAR <=*

euler_angle_convention: (required) *NX_CHAR <=*

axis_angle_convention: (required) *NX_CHAR <=*

orientation_parameterization_sign_convention: (required) *NX_CHAR <=*

processing_reference_frame: (required) *NXprocess <=*

reference_frame_type: (required) *NX_CHAR <=*

xaxis_direction: (required) *NX_CHAR <=*

xaxis_alias: (required) *NX_CHAR <=*

yaxis_direction: (required) *NX_CHAR <=*

yaxis_alias: (required) *NX_CHAR <=*

zaxis_direction: (required) *NX_CHAR <=*

zaxis_alias: (required) *NX_CHAR <=*

origin: (required) *NX_CHAR <=*

sample_reference_frame: (required) *NXprocess <=*

reference_frame_type: (required) *NX_CHAR <=*

xaxis_direction: (required) *NX_CHAR <=*

yaxis_direction: (required) *NX_CHAR <=*

zaxis_direction: (required) *NX_CHAR <=*

origin: (required) *NX_CHAR <=*

ROI_SET: (required) *NXcg_roi_set*

The simulated or characterized material volume element aka domain. At least one instance of geometry required either NXcg_grid, NXcg_polyhedron_set, or NXcg_point_set. This geometry group needs to contain details about the boundary conditions. **grid**: (required) *NXcg_grid*

dimensionality: (required) *NX_POSINT <=*

cardinality: (required) *NX_POSINT <=*

origin: (required) *NX_NUMBER <=*

symmetry: (required) *NX_CHAR <=*

cell_dimensions: (required) *NX_NUMBER <=*

extent: (required) *NX_POSINT <=*

identifier_offset: (required) *NX_INT <=*

boundary: (required) *NXcg_polyhedron_set*

A tight bounding box or sphere or bounding primitive about the grid.

number_of_boundaries: (required) *NX_POSINT* {units=*NX_UNITLESS*}

How many distinct boundaries are distinguished? Most grids discretize a cubic or cuboidal region. In this case six sides can be distinguished, each making an own boundary.

boundaries: (optional) *NX_CHAR*

Name of the boundaries. E.g. left, right, front, back, bottom, top, The field must have as many entries as there are number_of_boundaries.

boundary_conditions: (required) *NX_INT* (Rank: 1, Dimensions: [n_b]) {units=*NX_UNITLESS*}

The boundary conditions for each boundary:

0 - undefined 1 - open 2 - periodic 3 - mirror 4 - von Neumann 5 - Dirichlet

snapshot_set: (required) *NXms_snapshot_set*

Collection of microstructural data observed/simulated.

identifier_offset: (required) *NX_UINT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing snapshots. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

evolution: (optional) *NXprocess*

Summary quantities which are the result of some post-processing of the snapshot data (averaging, integrating, interpolating). Frequently used descriptors from continuum mechanics and thermodynamics can be used here. A few examples are given. Each descriptor is currently modelled as an instance of an NXprocess because it is relevant to understand how the descriptors are computed. **time**: (optional) *NXprocess*

Evolution of the physical time.

temperature: (optional) *NXprocess*

Evolution of the simulated temperature over time.

kinetics: (optional) *NXprocess*

Evolution of the recrystallized volume fraction over time.

MS_SNAPSHOT: (required) *NXms_snapshot* *<=*

time: (required) *NX_NUMBER* {units=*NX_TIME*} *<=*

Measured or simulated physical time stamp for this snapshot. Not to be confused with wall-clock timing or profiling data.

temperature: (required) *NX_NUMBER* {units=*NX_TEMPERATURE*}

Simulated temperature.

iteration: (required) *NX_UINT* {units=*NX_UNITLESS*}

Iteration or increment counter.

grid: (recommended) *NXcg_grid*

grain_identifier: (recommended) *NX_UINT* (Rank: 3, Dimensions: [n_x, n_y, n_z]) {units=*NX_UNITLESS*}

Grain identifier for each cell.

thread_identifier: (optional) *NX_UINT* (Rank: 3, Dimensions: [n_x, n_y, n_z]) {units=*NX_UNITLESS*}

Identifier of the OpenMP thread which processed this part of the grid.

recrystallization_front: (recommended) *NXprocess*

Details about those cells which in this time step represent the discretized recrystallization front. **halo_region**: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_front]) {units=*NX_UNITLESS*}

Which cells are currently in a halo region of threads.

mobility_weight: (recommended) *NX_NUMBER* (Rank: 1, Dimensions: [n_front]) {units=*NX_UNITLESS*}

So-called mobility weight which is a scaling factor to control the mobility of the grain boundary which is assumed to sweep currently this volume.

coordinate: (recommended) *NX_NUMBER* (Rank: 2, Dimensions: [n_front, 3]) {units=*NX_LENGTH*}

Grid coordinates of each cell in the recrystallization front.

deformed_grain_identifier: (recommended) *NX_UINT* (Rank: 1, Dimensions: [n_front]) {units=*NX_UNITLESS*}

Grain identifier assigned to each cell in the recrystallization front.

recrystallized_grain_identifier: (recommended) *NX_UINT* (Rank: 1, Dimensions: [n_front]) {units=*NX_UNITLESS*}

Grain identifier assigned to each nucleus which affected that cell in the recrystallization front.

recrystallized_volume_fraction: (recommended) *NX_NUMBER* (Rank: 1, Dimensions: [n_front]) {units=*NX_DIMENSIONLESS*}

Relative volume transformed as a measure of infection progress.

thread_identifier: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_front]) {units=*NX_UNITLESS*}

Identifier of the OpenMP thread processing each cell in the recrystallization front.

infection_direction: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_front]) {units=*NX_UNITLESS*}

Hint about the direction from which the cell was infected.

grain_ensemble: (recommended) *NXprocess*

Current grain-related quantities.

euler: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_grains, 3]) {units=*NX_ANGLE*}

Bunge-Euler angle triplets for each grain.

volume: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_grains]) {units=*NX_VOLUME*}

Discrete volume of each grain accounting also for partially transformed cells.

dislocation_density: (recommended) *NX_NUMBER* (Rank: 1, Dimensions: [n_grains]) {units=*NX_ANY*}

Current value for the dislocation density as a measure of the remaining stored energy in assumed crystal defects inside each grain. The difference between these values scales the driving force of grain boundary migration.

is_deformed: (recommended) *NX_BOOLEAN* (Rank: 1, Dimensions: [n_grains])

Is the grain deformed.

is_recrystallized: (recommended) *NX_BOOLEAN* (Rank: 1, Dimensions: [n_grains])

Is the grain recrystallized.

recrystallized_kinetics: (required) *NXprocess*

Current recrystallized volume fraction.

value: (required) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

Currently evaluated actual recrystallized volume fraction. This takes into account partially recrystallized cells.

target: (required) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

Currently desired target recrystallized volume fraction at which the user requested to log a snapshot.

stress: (optional) *NXprocess*

value: (required) *NX_NUMBER* (Rank: 2, Dimensions: [3, 3]) {units=*NX_ANY*}

Currently assumed globally applied Cauchy stress tensor on the ROI.

strain: (optional) *NXprocess*

value: (required) *NX_NUMBER* (Rank: 2, Dimensions: [3, 3]) {units=*NX_UNITLESS*}

Currently assumed globally applied Cauchy strain tensor on the ROI.

performance: (required) *NXcs_profiling*

- **current_working_directory**: (required) *NX_CHAR* <=
- **command_line_call**: (optional) *NX_CHAR* <=
- **start_time**: (recommended) *NX_DATE_TIME* <=
- **end_time**: (recommended) *NX_DATE_TIME* <=
- **total_elapsed_time**: (required) *NX_NUMBER* <=
- **number_of_processes**: (required) *NX_POSINT* <=
- **number_of_threads**: (required) *NX_POSINT* <=
- **number_of_gpus**: (required) *NX_POSINT* <=
- **CS_COMPUTER**: (recommended) *NXcs_computer* <=
  - **name**: (recommended) *NX_CHAR* <=
  - **operating_system**: (required) *NX_CHAR* <=
    - **@version**: (required) *NX_CHAR* <=
  - **uuid**: (optional) *NX_CHAR* <=
  - **CS_CPU**: (optional) *NXcs_cpu* <=
    - **name**: (optional) *NX_CHAR* <=
    - **FABRICATION**: (recommended) *NXfabrication* <=
      - **identifier**: (optional) *NX_CHAR* <=
      - **capabilities**: (optional) *NX_CHAR*
  - **CS_GPU**: (optional) *NXcs_gpu* <=
    - **name**: (optional) *NX_CHAR* <=
    - **FABRICATION**: (recommended) *NXfabrication* <=
      - **identifier**: (optional) *NX_CHAR* <=
      - **capabilities**: (optional) *NX_CHAR*
  - **CS_MM_SYS**: (optional) *NXcs_mm_sys* <=
    - **total_physical_memory**: (required) *NX_NUMBER* <=
  - **CS_IO_SYS**: (optional) *NXcs_io_sys* <=
    - **CS_IO_OBJ**: (required) *NXcs_io_obj* <=
      - **technology**: (required) *NX_CHAR* <=
      - **max_physical_capacity**: (required) *NX_NUMBER* <=
      - **name**: (optional) *NX_CHAR* <=
      - **FABRICATION**: (recommended) *NXfabrication* <=
        - **identifier**: (optional) *NX_CHAR* <=
        - **capabilities**: (optional) *NX_CHAR*
  - **CS_PROFILING_EVENT**: (required) *NXcs_profiling_event*

start_time: (optional) *NX_DATE_TIME <=*

end_time: (optional) *NX_DATE_TIME <=*

description: (required) *NX_CHAR <=*

elapsed_time: (required) *NX_NUMBER <=*

number_of_processes: (required) *NX_POSINT <=*

Specify if it was different from the number_of_processes in the NXcs_profiling super class.

number_of_threads: (required) *NX_POSINT <=*

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

number_of_gpus: (required) *NX_POSINT <=*

Specify if it was different from the number_of_threads in the NXcs_profiling super class.

max_virtual_memory_snapshot: (recommended) *NX_NUMBER <=*

max_resident_memory_snapshot: (recommended) *NX_NUMBER <=*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXms_score_results/ENTRY-group*
- */NXms_score_results/ENTRY/analysis_description-field*
- */NXms_score_results/ENTRY/analysis_identifier-field*
- */NXms_score_results/ENTRY/config_filename-field*
- */NXms_score_results/ENTRY/config_filename@version-attribute*
- */NXms_score_results/ENTRY/conventions-group*
- */NXms_score_results/ENTRY/conventions/processing_reference_frame-group*
- */NXms_score_results/ENTRY/conventions/processing_reference_frame/origin-field*
- */NXms_score_results/ENTRY/conventions/processing_reference_frame/reference_frame_type-field*
- */NXms_score_results/ENTRY/conventions/processing_reference_frame/xaxis_alias-field*
- */NXms_score_results/ENTRY/conventions/processing_reference_frame/xaxis_direction-field*
- */NXms_score_results/ENTRY/conventions/processing_reference_frame/yaxis_alias-field*
- */NXms_score_results/ENTRY/conventions/processing_reference_frame/yaxis_direction-field*
- */NXms_score_results/ENTRY/conventions/processing_reference_frame/zaxis_alias-field*
- */NXms_score_results/ENTRY/conventions/processing_reference_frame/zaxis_direction-field*
- */NXms_score_results/ENTRY/conventions/rotation_conventions-group*
- */NXms_score_results/ENTRY/conventions/rotation_conventions/axis_angle_convention-field*
- */NXms_score_results/ENTRY/conventions/rotation_conventions/euler_angle_convention-field*

- */NXms_score_results/ENTRY/conventions/rotation_conventions/orientation_parameterization_sign_convention-field*
- */NXms_score_results/ENTRY/conventions/rotation_conventions/rotation_convention-field*
- */NXms_score_results/ENTRY/conventions/rotation_conventions/three_dimensional_rotation_handedness-field*
- */NXms_score_results/ENTRY/conventions/sample_reference_frame-group*
- */NXms_score_results/ENTRY/conventions/sample_reference_frame/origin-field*
- */NXms_score_results/ENTRY/conventions/sample_reference_frame/reference_frame_type-field*
- */NXms_score_results/ENTRY/conventions/sample_reference_frame/xaxis_direction-field*
- */NXms_score_results/ENTRY/conventions/sample_reference_frame/yaxis_direction-field*
- */NXms_score_results/ENTRY/conventions/sample_reference_frame/zaxis_direction-field*
- */NXms_score_results/ENTRY/COORDINATE_SYSTEM_SET-group*
- */NXms_score_results/ENTRY/COORDINATE_SYSTEM_SET/TRANSFORMATIONS-group*
- */NXms_score_results/ENTRY/DATA-group*
- */NXms_score_results/ENTRY/definition-field*
- */NXms_score_results/ENTRY/end_time-field*
- */NXms_score_results/ENTRY/experiment_or_simulation-field*
- */NXms_score_results/ENTRY/performance-group*
- */NXms_score_results/ENTRY/performance/command_line_call-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER-group*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_CPU-group*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION-group*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/capabilities-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_CPU/FABRICATION/identifier-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_CPU/name-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_GPU-group*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION-group*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/capabilities-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_GPU/FABRICATION/identifier-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_GPU/name-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS-group*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ-group*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION-group*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/capabilities-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identifier-field*

- */NXms_score_results/ENTRY/ROI_SET/boundary/boundary_conditions-field*
- */NXms_score_results/ENTRY/ROI_SET/boundary/number_of_boundaries-field*
- */NXms_score_results/ENTRY/ROI_SET/grid-group*
- */NXms_score_results/ENTRY/ROI_SET/grid/cardinality-field*
- */NXms_score_results/ENTRY/ROI_SET/grid/cell_dimensions-field*
- */NXms_score_results/ENTRY/ROI_SET/grid/dimensionality-field*
- */NXms_score_results/ENTRY/ROI_SET/grid/extent-field*
- */NXms_score_results/ENTRY/ROI_SET/grid/identifier_offset-field*
- */NXms_score_results/ENTRY/ROI_SET/grid/origin-field*
- */NXms_score_results/ENTRY/ROI_SET/grid/symmetry-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set-group*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/evolution-group*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/evolution/kinetics-group*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/evolution/temperature-group*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/evolution/time-group*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/identifier_offset-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT-group*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/grain_ensemble-group*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/grain_ensemble/dislocation_density-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/grain_ensemble/euler-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/grain_ensemble/is_deformed-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/grain_ensemble/is_recrystallized-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/grain_ensemble/volume-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/grid-group*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/grid/grain_identifier-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/grid/thread_identifier-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/iteration-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/recrystallization_front-group*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/recrystallization_front/coordinate-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/recrystallization_front/deformed_grain_identifier-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/recrystallization_front/halo_region-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/recrystallization_front/infection_direction-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/recrystallization_front/mobility_weight-field*

- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/recrystallization_front/recrystallized_grain_identifier-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/recrystallization_front/recrystallized_volume_fraction-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/recrystallization_front/thread_identifier-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/recrystallized_kinetics-group*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/recrystallized_kinetics/target-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/recrystallized_kinetics/value-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/strain-group*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/strain/value-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/stress-group*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/stress/value-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/temperature-field*
- */NXms_score_results/ENTRY/ROI_SET/snapshot_set/MS_SNAPSHOT/time-field*
- */NXms_score_results/ENTRY/specimen-group*
- */NXms_score_results/ENTRY/specimen/name-field*
- */NXms_score_results/ENTRY/start_time-field*
- */NXms_score_results/ENTRY/status-field*
- */NXms_score_results/ENTRY/USER-group*
- */NXms_score_results/ENTRY/USER/address-field*
- */NXms_score_results/ENTRY/USER/affiliation-field*
- */NXms_score_results/ENTRY/USER/email-field*
- */NXms_score_results/ENTRY/USER/name-field*
- */NXms_score_results/ENTRY/USER/orcid-field*
- */NXms_score_results/ENTRY/USER/orcid_platform-field*
- */NXms_score_results/ENTRY/USER/role-field*
- */NXms_score_results/ENTRY/USER/social_media_name-field*
- */NXms_score_results/ENTRY/USER/social_media_platform-field*
- */NXms_score_results/ENTRY/USER/telephone_number-field*
- */NXms_score_results/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXms_score_results.nxdl.xml

NXms_snapshot

Status:

base class, extends *NXobject*

Description:

Base class for data on the state of the microstructure at a given time/iteration.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
none

Structure:

comment: (optional) *NX_CHAR*

Is this time for a measurement or a simulation.

Any of these values: `measurement` | `simulation`

time: (optional) *NX_NUMBER* {units=*NX_TIME*}

Measured or simulated physical time stamp for this snapshot. Not to be confused with wall-clock timing or profiling data.

iteration: (optional) *NX_INT* {units=*NX_UNITLESS*}

Iteration or increment counter.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXms_snapshot/comment-field*
- */NXms_snapshot/iteration-field*
- */NXms_snapshot/time-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXms_snapshot.nxdl.xml

NXms_snapshot_set

Status:

base class, extends *NXobject*

Description:

A collection of spatiotemporal microstructure data.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
NXms_snapshot

Structure:

comment: (optional) *NX_CHAR*

Is this set describing a measurement or a simulation?

Any of these values: `measurement` | `simulation`

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing snapshots. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

MS_SNAPSHOT: (optional) *NXms_snapshot*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXms_snapshot_set/comment-field*
- */NXms_snapshot_set/identifier_offset-field*
- */NXms_snapshot_set/MS_SNAPSHOT-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXms_snapshot_set.nxdl.xml

NXopt

Status:

application definition, extends *NXobject*

Description:

An application definition for optical spectroscopy experiments.

Symbols:

Variables used throughout the document, e.g. dimensions or parameters.

N_spectrum: Length of the spectrum array (e.g. wavelength or energy) of the measured data.

N_sensors: Number of sensors used to measure parameters that influence the sample, such as temperature or pressure.

N_measurements: Number of measurements (1st dimension of measured_data array). This is equal to the number of parameters scanned. For example, if the experiment was performed at three different temperatures and two different pressures N_measurements = 2*3 = 6.

N_detection_angles: Number of detection angles of the beam reflected or scattered off the sample.

N_incident_angles: Number of angles of incidence of the incident beam.

N_observables: Number of observables that are saved in a measurement. e.g. one for intensity, reflectivity or transmittance, two for Psi and Delta etc. This is equal to the second dimension of the data array 'measured_data' and the number of column names.

Groups cited:

NXaperture, *NXbeam_path*, *NXdata*, *NXentry*, *NXenvironment*, *NXinstrument*, *NXprocess*, *NXprogram*, *NXsample*, *NXsensor*, *NXsubentry*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

An application definition template for optical spectroscopy experiments.

A general optical experiment consists of a light or excitation source, a beam path, a sample + its stage + its environment, and a detection unit. Examples are reflection or transmission measurements, photoluminescence, Raman spectroscopy, ellipsometry etc. **definition**: (required) *NX_CHAR <=*

An application definition describing a general optical experiment.

Obligatory value: `NXopt` **@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 <=*

A (globally persistent) unique identifier of the experiment. (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: (optional) *NX_CHAR <=*

An optional free-text description of the experiment.

However, details of the experiment should be defined in the specific fields of this application definition rather than in this experiment description.

experiment_type: (required) *NX_CHAR*

Specify the type of the optical experiment.

start_time: (required) *NX_DATE_TIME <=*

Start time of the experiment. UTC offset should be specified.

USER: (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: (recommended) *NX_CHAR <=*

Name of the affiliation of the user at the point in time when the experiment was performed.

address: (recommended) *NX_CHAR <=*

Street address 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* <=

Telephone number of the user.

INSTRUMENT: (required) *NXinstrument* <=

Properties of the experimental setup. This includes general information about the instrument (such as model, company etc.), information about the calibration of the instrument, elements of the beam path including the excitation or light source and the detector unit, the sample stage (plus the sample environment, which also includes sensors used to monitor external conditions) and elements of the beam path.

Meta data describing the sample should be specified in ENTRY/SAMPLE outside of ENTRY/INSTRUMENT. **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.

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 ENTRY/INSTRUMENT/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_incident_angles]) {units=*NX_ANGLE*}

Angle(s) of the incident beam vs. the normal of the bottom reflective (substrate) surface in the sample. **@units**: (required) *NX_CHAR*

detection_angle: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [N_detection_angles]) {units=*NX_ANGLE*}

Detection angle(s) of the beam reflected or scattered off the sample vs. the normal of the bottom reflective (substrate) surface in the sample if not equal to the angle(s) of incidence.

software: (required) *NXprocess*

@url: (optional) *NX_CHAR*

Website of the software.

program: (required) *NX_CHAR <=*

Commercial or otherwise defined given name of the program that was used to measure the data, i.e. the software used to start and record the measured data and/or metadata. 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.

firmware: (recommended) *NXprogram*

Commercial or otherwise defined name of the firmware that was used for the measurement - if available. **@version**: (required) *NX_CHAR*

Version and build number or commit hash of the software source code.

@url: (optional) *NX_CHAR*

Website of the software.

calibration: (recommended) *NXsubentry*

The calibration data and metadata should be described in a separate NeXus file with the link provided in 'calibration_link'. **calibration_time**: (optional) *NX_DATE_TIME*

If calibtration status is 'calibration time provided', specify the ISO8601 date when calibration was last performed before this measurement. UTC offset should be specified.

calibration_data_link: (required) *NX_CHAR*

Link to the NeXus file containing the calibration data and metadata.

BEAM_PATH: (required) *NXbeam_path*

Describes an arrangement of optical or other elements, e.g. the beam path between the light source and the sample, or between the sample and the detector unit (including the sources and detectors themselves).

If a beam splitter (i.e. a device that splits the incoming beam into two or more beams) is part of the beam path, two or more NXbeam_path fields may be needed to fully describe the beam paths and the correct sequence of the beam path elements. Use as many beam paths as needed to describe the setup.

sample_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). **stage_type**: (required) *NX_CHAR*

Specify the type of the sample stage.

Any of these values:

- `manual stage`
- `scanning stage`
- `liquid stage`
- `gas cell`
- `cryostat`

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 the plane of incidence'.

environment_conditions: (required) *NXenvironment*

Specify external parameters that have influenced the sample, such as the surrounding medium, and varied parameters e.g. temperature, pressure, pH value, optical excitation etc. **medium**: (required) *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) *NX_FLOAT* (Rank: 2, Dimensions: [2, N_spectrum]) {units=*NX_UNITLESS*}

Array of pairs of complex refractive indices n + ik of the medium for every measured spectral point/wavelength/energy. Only necessary if the measurement was performed not in air, or something very well known, e.g. high purity water.

PARAMETER: (optional) *NXsensor <=*

A sensor used to monitor an external condition influencing the sample, such as temperature or pressure. It is suggested to replace 'PARAMETER' by the type of the varied parameter defined in 'parameter_type'. The measured parameter values should be provided in 'values'. For each parameter, a 'PARAMETER(NXsensor)' field needs to exist. In other words, there are N_sensors 'PARAMETER(NXsensor)' fields. **parameter_type**: (required) *NX_CHAR*

Indicates which parameter was changed. Its definition must exist below. The specified variable has to be N_measurements long, providing the parameters for each data set. If you vary more than one parameter simultaneously. If the measured parameter is not contained in the list *other* should be specified and the *parameter_type_name* should be provided.

Any of these values:

- `conductivity`
- `detection_angle`
- `electric_field`
- `flow`
- `incident_angle`

- `magnetic_field`
- `optical_excitation`
- `pH`
- `pressure`
- `resistance`
- `shear`
- `stage_positions`
- `strain`
- `stress`
- `surface_pressure`
- `temperature`
- `voltage`
- `other`

parameter_type_name: (optional) *NX_CHAR*

If the parameter_type is *other* a name should be specified here.

number_of_parameters: (required) *NX_POSINT* {units=*NX_UNITLESS*}

Number of different parameter values at which the measurement was performed. For example, if the measurement was performed at temperatures of 4, 77 and 300 K, then number_of_parameters = 3.

values: (required) *NX_FLOAT* (Rank: 1, Dimensions: [N_measurements]) {units=*NX_ANY*}

Vector containing the values of the varied parameter. Its length is equal to N_measurements. The order of the values should be as follows:

- Order the sensors according to number_of_parameters starting with the lowest number. If number_of_parameters is equal for two sensors order them alphabetically (sensor/parameter name).
- The first sensor's j parameters should be ordered in the following way. The first N_measurements/number_of_parameters entries of the vector contain the first parameter (a1), the second N_measurements/number_of_parameters contain the second parameter (a2) etc., so the vector looks like: [a1,a1,…,a1, a2,a2,…,a2, … aj,aj,…aj]
- The kth sensor's m parameters should be ordered in the following way: [p1,…p1,p2,…,p2,…pm,…,pm, p1,…p1,p2,…,p2,…pm,…,pm, … p1,…p1,p2,…,p2,…pm,…,pm]
- The last sensor's n parameters should be ordered in the following way: [z1,z2,…,zn, z1,z2,…,zn, … z1,z2,…,zn]

For example, if the experiment was performed at three different temperatures (T1, T2, T3), two different pressures (p1, p2) and two different angles of incidence (a1, a2), then N_measurements = 12 and the order of the values for the various parameter vectors is:

- angle_of_incidence: [a1,a1,a1,a1,a1,a1,a2,a2,a2,a2,a2,a2]
- pressure: [p1,p1,p1,p2,p2,p2,p1,p1,p1,p2,p2,p2]
- temperature: [T1,T2,T3,T1,T2,T3,T1,T2,T3,T1,T2,T3]

WINDOW: (optional) *NXaperture*

For environmental measurements, the environment (liquid, vapor etc.) is enclosed in a cell, which has windows both in the direction of the source (entry window) and the detector (exit window) (looking from the sample). In case that the entry and exit windows are not the same type and do not have the same properties, use a second 'WINDOW(MXaperture)' field.

The 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 (reference sample) or the actual sample of interest in the environmental cell. State if a window correction has been performed in 'window_effects_corrected'. Reference data should be considered as a separate experiment, and a link to the NeXus file should be added in reference_data_link in measured_data.

The window is considered to be a part of the sample stage but also beam path. Hence, its position within the beam path should be defined by the 'depends_on' field. **depends_on**: (recommended) *NX_CHAR <=*

Specify the position of the window in the beam path by pointing to the preceding element in the sequence of beam path elements.

window_effects_corrected: (required) *NX_BOOLEAN*

Was a window correction performed? If 'True' describe the window correction procedure in 'window_correction/procedure'.

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 material, decsribe here what it is.

thickness: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Thickness of the window.

orientation_angle: (required) *NX_FLOAT* {units=*NX_ANGLE*}

Angle of the window normal (outer) vs. the substrate normal (similar to the angle of incidence).

window_correction: (optional) *NXprocess*

Was a window correction performed? If 'True' describe the window correction procedure in '' **procedure**: (required) *NX_CHAR*

Describe when (before or after the main measurement + time stamp in 'date') and how the window effects have been corrected, i.e. either mathematically or by performing a reference measurement. In the latter case, provide the link to to the reference data in 'reference_data_link'.

reference_data_link: (optional) *NX_CHAR*

Link to the NeXus file which describes the reference data if a reference measurement for window correction was performed. Ideally, the reference measurement was performed on a reference sample and on the same sample, and using the same conditions as for the actual measurement with and without windows. It should have been conducted as close in time to the actual measurement as possible.

SAMPLE: (required) *NXsample* <=

Properties of the sample, such as sample type, layer structure, chemical formula, atom types, its history etc. Information about the sample stage and sample environment should be described in ENTRY/INSTRUMENT/sample_stage. **sample_name**: (required) *NX_CHAR*

Descriptive name of the sample

sample_type: (required) *NX_CHAR*

Specify the type of sample, e.g. thin film, single crystal etc.

Any of these values:

- `thin film`
- `single crystal`
- `poly crystal`
- `single layer`
- `multi layer`

layer_structure: (required) *NX_CHAR*

Qualitative description of the layer structure for the sample, starting with the top layer (i.e. the one on the front surface, on which the light incident), e.g. native oxide/bulk substrate, or Si/native oxide/thermal oxide/polymer/peptide.

chemical_formula: (required) *NX_CHAR* <=

Chemical formula of the sample. Use the Hill system (explained here: https://en.wikipedia.org/wiki/Chemical_formula#Hill_system) to write the chemical formula. In case the sample consists of several layers, this should be a list of the chemical formulas of the individual layers, where the first entry is the chemical formula of the top layer (the one on the front surface, on which the light incident). The order must be consistent with layer_structure

atom_types: (required) *NX_CHAR*

List of comma-separated elements from the periodic table that are contained in the sample. If the sample substance has multiple components, all elements from each component must be included in 'atom_types'.

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.

substrate: (recommended) *NX_CHAR*

Description of the substrate.

sample_orientation: (optional) *NX_CHAR*

Specify the sample orientation.

data_collection: (required) *NXprocess* *<=*

Measured data, data errors, and varied parameters. If reference data were measured they should be considered a separate experiment and a link to its NeXus file should be added in reference_data_link. **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 intensity, reflectivity, transmittance, Psi and Delta etc. It is possible to have multiple selections. The enumeration list depends on the type of experiment and may differ for different application definitions.

Any of these values:

- `intensity`
- `reflectivity`
- `transmittance`
- `Psi/Delta`
- `tan(Psi)/cos(Delta)`
- `Mueller matrix`
- `Jones matrix`

- `N/C/S`
- `raw data`

NAME_spectrum: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [N_spectrum]) {units=*NX_ANY*}

Spectral values (e.g. wavelength or energy) used for the measurement. An array of 1 or more elements. Length defines N_spectrum. Replace 'SPECTRUM' by the physical quantity that is used, e.g. wavelength. **@units**: (optional) *NX_CHAR*

If applicable, change 'unit: NX_ANY' to the appropriate NXDL unit. If the unit of the measured data is not covered by NXDL units state here which unit was used.

measured_data: (required) *NX_FLOAT* (Rank: 3, Dimensions: [N_measurements, N_observables, N_spectrum]) {units=*NX_ANY*}

Resulting data from the measurement, described by 'data_type'.

The first dimension is defined by the number of measurements taken, (N_measurements). The instructions on how to order the values contained in the parameter vectors given in the doc string of INSTRUMENT/sample_stage/environment_conditions/PARAMETER/values, define the N_measurements parameter sets. For example, if the experiment was performed at three different temperatures (T1, T2, T3), two different pressures (p1, p2) and two different angles of incidence (a1, a2), the first measurement was taken at the parameters {a1,p1,T1}, the second measurement at {a1,p1,T2} etc. **@units**: (optional) *NX_CHAR*

If applicable, change 'unit: NX_ANY' to the appropriate NXDL unit. If the unit of the measured data is not covered by NXDL units state here which unit was used.

measured_data_errors: (optional) *NX_FLOAT* (Rank: 3, Dimensions: [N_measurements, N_observables, N_spectrum]) {units=*NX_ANY*}

Specified uncertainties (errors) of the data described by 'data_type' and provided in 'measured_data'. **@units**: (optional) *NX_CHAR*

If applicable, change 'unit: NX_ANY' to the appropriate NXDL unit. If the unit of the measured data is not covered by NXDL units state here which unit was used.

varied_parameter_link: (optional) *NX_CHAR* (Rank: 1, Dimensions: [N_sensors])

List of links to the values of the sensors. Add a link for each varied parameter (i.e. for each sensor).

reference_data_link: (optional) *NX_CHAR*

Link to the NeXus file which describes the reference data if a reference measurement was performed. Ideally, the reference measurement was performed using the same conditions as the actual measurement and should be as close in time to the actual measurement as possible.

data_software: (optional) *NXprocess*

@url: (optional) *NX_CHAR*

Website of the software.

program: (required) *NX_CHAR* *<=*

Commercial or otherwise defined given name of the program that was used to generate the result file(s) with measured data and/or metadata (in most cases, this is the same as INSTRUMENT/software). 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.

DATA: (optional) *NXdata*

A plot of the multi-dimensional data array provided in ENTRY/data/measured_data. **@axes**: (required) *NX_CHAR* *<=*

Spectrum, i.e. x-axis of the data (e.g. wavelength, energy etc.)

derived_parameters: (optional) *NXprocess* *<=*

Parameters that are derived from the measured data.

depolarization: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [N_measurements, 1, N_spectrum]) {units=*NX_UNITLESS*}

Light loss due to depolarization as a value in [0-1].

Jones_quality_factor: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [N_measurements, 1, N_spectrum]) {units=*NX_UNITLESS*}

Jones quality factor.

reflectivity: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [N_measurements, 1, N_spectrum]) {units=*NX_UNITLESS*}

Reflectivity.

transmittance: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [N_measurements, 1, N_spectrum]) {units=*NX_UNITLESS*}

Transmittance.

ANALYSIS_program: (optional) *NXprocess*

program: (required) *NX_CHAR* *<=*

Commercial or otherwise defined given name of the program that was used to generate or calculate the derived parameters. 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.

plot: (required) *NXdata* *<=*

A default view of the data provided in ENTRY/data_collection/measured_data. This should be the part of the data set which provides the most suitable representation of the data. **@axes**: (required) *NX_CHAR* *<=*

Spectrum, i.e. x-axis of the data (e.g. wavelength, energy etc.)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXopt/ENTRY-group*
- */NXopt/ENTRY/data_collection-group*
- */NXopt/ENTRY/data_collection/DATA-group*
- */NXopt/ENTRY/data_collection/DATA@axes-attribute*
- */NXopt/ENTRY/data_collection/data_identifier-field*
- */NXopt/ENTRY/data_collection/data_software-group*
- */NXopt/ENTRY/data_collection/data_software/program-field*
- */NXopt/ENTRY/data_collection/data_software/version-field*
- */NXopt/ENTRY/data_collection/data_software@url-attribute*
- */NXopt/ENTRY/data_collection/data_type-field*
- */NXopt/ENTRY/data_collection/measured_data-field*
- */NXopt/ENTRY/data_collection/measured_data@units-attribute*
- */NXopt/ENTRY/data_collection/measured_data_errors-field*
- */NXopt/ENTRY/data_collection/measured_data_errors@units-attribute*
- */NXopt/ENTRY/data_collection/NAME_spectrum-field*
- */NXopt/ENTRY/data_collection/NAME_spectrum@units-attribute*
- */NXopt/ENTRY/data_collection/reference_data_link-field*
- */NXopt/ENTRY/data_collection/varied_parameter_link-field*
- */NXopt/ENTRY/definition-field*
- */NXopt/ENTRY/definition@url-attribute*
- */NXopt/ENTRY/definition@version-attribute*
- */NXopt/ENTRY/derived_parameters-group*
- */NXopt/ENTRY/derived_parameters/ANALYSIS_program-group*
- */NXopt/ENTRY/derived_parameters/ANALYSIS_program/program-field*
- */NXopt/ENTRY/derived_parameters/ANALYSIS_program/version-field*
- */NXopt/ENTRY/derived_parameters/depolarization-field*
- */NXopt/ENTRY/derived_parameters/Jones_quality_factor-field*
- */NXopt/ENTRY/derived_parameters/reflectivity-field*
- */NXopt/ENTRY/derived_parameters/transmittance-field*
- */NXopt/ENTRY/experiment_description-field*
- */NXopt/ENTRY/experiment_identifier-field*
- */NXopt/ENTRY/experiment_type-field*

- /NXopt/ENTRY/INSTRUMENT-group
- /NXopt/ENTRY/INSTRUMENT/angle_of_incidence-field
- /NXopt/ENTRY/INSTRUMENT/angle_of_incidence@units-attribute
- /NXopt/ENTRY/INSTRUMENT/BEAM_PATH-group
- /NXopt/ENTRY/INSTRUMENT/calibration-group
- /NXopt/ENTRY/INSTRUMENT/calibration/calibration_data_link-field
- /NXopt/ENTRY/INSTRUMENT/calibration/calibration_time-field
- /NXopt/ENTRY/INSTRUMENT/calibration_status-field
- /NXopt/ENTRY/INSTRUMENT/company-field
- /NXopt/ENTRY/INSTRUMENT/construction_year-field
- /NXopt/ENTRY/INSTRUMENT/detection_angle-field
- /NXopt/ENTRY/INSTRUMENT/firmware-group
- /NXopt/ENTRY/INSTRUMENT/firmware@url-attribute
- /NXopt/ENTRY/INSTRUMENT/firmware@version-attribute
- /NXopt/ENTRY/INSTRUMENT/model-field
- /NXopt/ENTRY/INSTRUMENT/model@version-attribute
- /NXopt/ENTRY/INSTRUMENT/sample_stage-group
- /NXopt/ENTRY/INSTRUMENT/sample_stage/alternative-field
- /NXopt/ENTRY/INSTRUMENT/sample_stage/environment_conditions-group
- /NXopt/ENTRY/INSTRUMENT/sample_stage/environment_conditions/medium-field
- /NXopt/ENTRY/INSTRUMENT/sample_stage/environment_conditions/medium_refractive_indices-field
- /NXopt/ENTRY/INSTRUMENT/sample_stage/environment_conditions/PARAMETER-group
- /NXopt/ENTRY/INSTRUMENT/sample_stage/environment_conditions/PARAMETER/number_of_parameters-field
- /NXopt/ENTRY/INSTRUMENT/sample_stage/environment_conditions/PARAMETER/parameter_type-field
- /NXopt/ENTRY/INSTRUMENT/sample_stage/environment_conditions/PARAMETER/parameter_type_name-field
- /NXopt/ENTRY/INSTRUMENT/sample_stage/environment_conditions/PARAMETER/values-field
- /NXopt/ENTRY/INSTRUMENT/sample_stage/stage_type-field
- /NXopt/ENTRY/INSTRUMENT/sample_stage/WINDOW-group
- /NXopt/ENTRY/INSTRUMENT/sample_stage/WINDOW/depends_on-field
- /NXopt/ENTRY/INSTRUMENT/sample_stage/WINDOW/material-field
- /NXopt/ENTRY/INSTRUMENT/sample_stage/WINDOW/orientation_angle-field
- /NXopt/ENTRY/INSTRUMENT/sample_stage/WINDOW/other_material-field
- /NXopt/ENTRY/INSTRUMENT/sample_stage/WINDOW/thickness-field
- /NXopt/ENTRY/INSTRUMENT/sample_stage/WINDOW/window_correction-group
- /NXopt/ENTRY/INSTRUMENT/sample_stage/WINDOW/window_correction/procedure-field

- */NXopt/ENTRY/INSTRUMENT/sample_stage/WINDOW/window_correction/reference_data_link-field*
- */NXopt/ENTRY/INSTRUMENT/sample_stage/WINDOW/window_effects_corrected-field*
- */NXopt/ENTRY/INSTRUMENT/software-group*
- */NXopt/ENTRY/INSTRUMENT/software/program-field*
- */NXopt/ENTRY/INSTRUMENT/software/version-field*
- */NXopt/ENTRY/INSTRUMENT/software@url-attribute*
- */NXopt/ENTRY/plot-group*
- */NXopt/ENTRY/plot@axes-attribute*
- */NXopt/ENTRY/SAMPLE-group*
- */NXopt/ENTRY/SAMPLE/atom_types-field*
- */NXopt/ENTRY/SAMPLE/chemical_formula-field*
- */NXopt/ENTRY/SAMPLE/layer_structure-field*
- */NXopt/ENTRY/SAMPLE/preparation_date-field*
- */NXopt/ENTRY/SAMPLE/sample_history-field*
- */NXopt/ENTRY/SAMPLE/sample_name-field*
- */NXopt/ENTRY/SAMPLE/sample_orientation-field*
- */NXopt/ENTRY/SAMPLE/sample_type-field*
- */NXopt/ENTRY/SAMPLE/substrate-field*
- */NXopt/ENTRY/start_time-field*
- */NXopt/ENTRY/USER-group*
- */NXopt/ENTRY/USER/address-field*
- */NXopt/ENTRY/USER/affiliation-field*
- */NXopt/ENTRY/USER/email-field*
- */NXopt/ENTRY/USER/name-field*
- */NXopt/ENTRY/USER/orcid-field*
- */NXopt/ENTRY/USER/telephone_number-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXopt.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*}

Citing the JEOL TEM glossary this is *an effective distance from a specimen to a plane where an observed diffraction pattern is formed*.

magnification: (optional) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

The factor of enlargement of the apparent size, not physical size, of an object.

defocus: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The defocus aberration constant oftentimes taken as the C_1_0 which is described in more details in NXaberration.

semi_convergence_angle: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Citing the JEOL TEM glosssary this is the value *when a cone shaped, convergent electron beam illuminates a specimen, the semi-angle of the cone is termed convergence angle.*

field_of_view: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The extent of the observable parts of the specimen given the current magnification and other settings of the instrument.

working_distance: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Citing Globalsino this is *a distance between the specimen and the lower pole piece in SEM system*.

beam_current: (optional) *NX_FLOAT* {units=*NX_CURRENT*}

Beam current as measured relevant for the illumination of the specimen. Users should specify further details like how the beam current was measured using the beam_current_description field.

beam_current_description: (optional) *NX_CHAR*

Specify further details how the beam current was measured or estimated.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXoptical_system_em/beam_current-field*
- */NXoptical_system_em/beam_current_description-field*
- */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

NXorientation_set

Status:

base class, extends *NXobject*

Description:

Details about individual orientations of a set of objects.

For a more detailed insight into the discussion of parameterizing orientations in materials science see:

- https://doi.org/10.1016/j.matchar.2016.04.008
- https://doi.org/10.1088/0965-0393/23/8/083501
- https://doi.org/10.1007/978-3-662-09156-2 group-theory of rotations
- https://doi.org/10.1016/C2013-0-11769-2 the classical book of H.-J. Bunge

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

d: The dimensionality of the reference space/coordinate system.

c: The cardinality of the set, i.e. the number of orientations.

n_p: Number of parameters for the chosen parameterization.

Groups cited:
NXtransformations

Structure:

parameterization: (optional) *NX_CHAR*

Any of these values: `bunge-euler (ZXZ)` | `quaternion`

objects: (optional) *NX_CHAR*

A link or reference to the objects whose identifier are referred to in identifier to resolve which row tuple is the orientation of each object by reading orientations.

identifier_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies which orientation (row of array orientation) matches to which object.e first index to be used for distinguishing hexahedra. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval [identifier_offset, identifier_offset+c-1]. For explicit indexing the identifier array has to be defined.

The identifier_offset field can for example be used to communicate if the identifiers are expected to start from 1 (referred to as Fortran-/Matlab-) or from 0 (referred to as C-, Python-style index notation) respectively.

identifier: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

Integer used to distinguish how a row in orientation describes a specific object with an explicit identifier that can be queried via inspecting the list of available objects in objects.

The rational behind having such a more complicated pattern is that not all objects referred when following the link in objects may still exists or are still tracked when the orientation set was characterized.

This design enables to also use NXorientation_set in situations where the orientation of objects change as a function in time.

orientation: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, n_p]) {units=*NX_ANY*}

Parameterized orientations.

TRANSFORMATIONS: (optional) *NXtransformations*

Reference to or definition of a coordinate system with which the definitions are interpretable.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXorientation_set/identifier-field*
- */NXorientation_set/identifier_offset-field*
- */NXorientation_set/objects-field*
- */NXorientation_set/orientation-field*
- */NXorientation_set/parameterization-field*
- */NXorientation_set/TRANSFORMATIONS-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXorientation_set.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

NXpid

Status:

base class, extends *NXobject*

Description:

Contains the settings of a PID controller.

Symbols:

No symbol table

Groups cited:
NXlog, *NXsensor*

Structure:

description: (optional) *NX_CHAR*

Description of how the Process Value for the PID controller is produced by sensor(s) in the setup.

For example, a set of sensors could be averaged over before feeding it back into the loop.

setpoint: (optional) *NX_FLOAT* {units=*NX_ANY*}

The Setpoint(s) used as an input for the PID controller.

It can also be a link to an NXsensor.value field.

K_p_value: (optional) *NX_NUMBER*

Proportional term. The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant Kp, called the proportional gain constant.

K_i_value: (optional) *NX_NUMBER*

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain (Ki) and added to the controller output.

K_d_value: (optional) *NX_NUMBER*

The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain K_d. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, K_d

pv_sensor: (optional) *NXsensor*

The sensor representing the Process Value used in the feedback loop for the PID.

In case multiple sensors were used, this NXsensor should contain the proper calculated/aggregated value. **value_log**: (optional) *NXlog* <=

value: (optional) *NX_NUMBER* <=

The actual timeseries data fed back to the PID loop.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXpid/description-field*
- */NXpid/K_d_value-field*
- */NXpid/K_i_value-field*
- */NXpid/K_p_value-field*
- */NXpid/pv_sensor-group*
- */NXpid/pv_sensor/value_log-group*
- */NXpid/pv_sensor/value_log/value-field*
- */NXpid/setpoint-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXpid.nxdl.xml

NXpolarizer_opt

Status:

base class, extends *NXobject*

Description:

An optical polarizer.

Information on the properties of polarizer is provided e.g. [here](https://www.rp-photonics.com/polarizers.html).

Symbols:

N_spectrum: Size of the wavelength array for which the refractive index of the material and/or coating is given.

N_spectrum_RT: Size of the wavelength array for which the reflectance or transmission of the polarizer is given.

Groups cited:
NXdata, *NXsample*, *NXshape*

Structure:

type: (optional) *NX_CHAR*

Type of the polarizer (e.g. dichroic, linear, circular etc.)

Any of these values:

- `dichroic`
- `linear`
- `circular`
- `Glan-Thompson prism`
- `Nicol prism`
- `Glan-Taylor prism`
- `Glan-Focault prism`
- `Wollaston prism`
- `Normarski prism`
- `Senarmont prism`
- `thin-film polarizer`
- `wire grid polarizer`
- `other`

type_other: (optional) *NX_CHAR*

If you selected 'other' in type specify what it is.

polarizer_angle: (recommended) *NX_NUMBER* {units=*NX_ANGLE*}

Angle of the polarizer.

acceptance_angle: (recommended) *NX_NUMBER* (Rank: 1, Dimensions: [2]) {units=*NX_ANGLE*}

Acceptance angle of the polarizer (range).

wavelength_range: (recommended) *NX_FLOAT* (Rank: 1, Dimensions: [2]) {units=*NX_WAVELENGTH*}

Wavelength range for which the polarizer is designed. Enter the minimum and maximum wavelength (lower and upper limit) of the range.

extinction_ratio: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [N_spectrum]) {units=*NX_UNITLESS*}

Extinction ratio (maximum to minimum transmission).

reflection: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [N_spectrum_RT]) {units=*NX_UNITLESS*}

Reflection of the polarizer at given wavelength values.

transmission: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [N_spectrum_RT]) {units=*NX_UNITLESS*}

Transmission of the polarizer at given wavelength values.

SHAPE: (recommended) *NXshape*

Describe the geometry (shape, dimension etc.) of the device. Specify the dimensions in 'SHAPE/size'. A sketch of the device should be provided in the 'sketch(NXdata)' field to clarify (i) the shape and dimensions of the device, and (ii) the input and outputs (i.e. the direction of the incoming and outcoming (split) beams). **shape**: (optional) *NX_CHAR* *<=*

Describe the shape (plate, cube, wedged, prism etc.).

Any of these values:

- `cube`
- `cylinder`
- `plate`
- `prism`
- `wedged`
- `other`

other_shape: (optional) *NX_CHAR*

If you chose 'other' in 'shape' describe what it is.

size: (optional) *NX_CHAR* (Rank: 2, Dimensions: [N_objects, N_shapepar])

Physical extent of the device. The device might be made up of one or more objects (NX_objects). The meaning and location of the axes used will vary according to the value of the 'shape' variable. 'N_shapepar' defines how many parameters:

- For 'cube' the parameters are (width, length).
- For 'cylinder' the parameters are (diameter, length).
- For 'plate' the parameters are (width, height, length).
- For 'prism' the parameters are (width, height, length).
- For 'wedged' the parameters are (width, height, shortest length). The wedge angle should be provided in 'SHAPE/wedge_angle'.
- For 'other' the parameters may be (A, B, C, …) with the labels defined in the sketch plotted in 'SHAPE/sketch'.

wedge_angle: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

Wedge angle if 'shape' is 'wedged'.

sketch: (optional) *NXdata*

Sketch of thedevice showing its geometry. The paths of the incoming and outgoing beam should be illustrated and labelled (0 for the incoming beam, and 1, 2,…, N_outputs for the outputs).

substrate: (optional) *NXsample*

Properties of the substrate material of the polarizer. If the device has a coating specify the coating material and its properties in 'coating'. **substrate_material**: (optional) *NX_CHAR*

Specify the substrate material of the polarizer.

substrate_thickness: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Thickness of the polarizer substrate.

index_of_refraction: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [2, N_spectrum]) {units=*NX_UNITLESS*}

Complex index of refraction of the polarizer material. Specify at given spectral values (wavelength, energy, wavenumber etc.).

COATING: (optional) *NXsample*

If the device has a coating describe the material and its properties. Some basic information can be found e.g. [here] (https://www.opto-e.com/basics/reflection-transmission-and-coatings). If the back and front side of the polarizer are coated with different materials, you may define two coatings (e.g. COATING1 and COATING2). **coating_type**: (optional) *NX_CHAR*

Specify the coating type (e.g. dielectric, anti-reflection (AR), multilayer coating etc.).

coating_material: (optional) *NX_CHAR*

Describe the coating material (e.g. MgF2).

coating_thickness: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Thickness of the coating.

index_of_refraction_coating: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [2, N_spectrum_coating]) {units=*NX_UNITLESS*}

Complex index of refraction of the coating. Specify at given spectral values (wavelength, energy, wavenumber etc.).

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXpolarizer_opt/acceptance_angle-field*
- */NXpolarizer_opt/COATING-group*
- */NXpolarizer_opt/COATING/coating_material-field*
- */NXpolarizer_opt/COATING/coating_thickness-field*
- */NXpolarizer_opt/COATING/coating_type-field*
- */NXpolarizer_opt/COATING/index_of_refraction_coating-field*
- */NXpolarizer_opt/extinction_ratio-field*

- */NXpolarizer_opt/polarizer_angle-field*
- */NXpolarizer_opt/reflection-field*
- */NXpolarizer_opt/SHAPE-group*
- */NXpolarizer_opt/SHAPE/other_shape-field*
- */NXpolarizer_opt/SHAPE/shape-field*
- */NXpolarizer_opt/SHAPE/size-field*
- */NXpolarizer_opt/SHAPE/sketch-group*
- */NXpolarizer_opt/SHAPE/wedge_angle-field*
- */NXpolarizer_opt/substrate-group*
- */NXpolarizer_opt/substrate/index_of_refraction-field*
- */NXpolarizer_opt/substrate/substrate_material-field*
- */NXpolarizer_opt/substrate/substrate_thickness-field*
- */NXpolarizer_opt/transmission-field*
- */NXpolarizer_opt/type-field*
- */NXpolarizer_opt/type_other-field*
- */NXpolarizer_opt/wavelength_range-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXpolarizer_opt.nxdl.xml

NXprogram

Status:

base class, extends *NXobject*

Description:

Base class to describe a software tool or library.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
none

Structure:

program: (optional) *NX_CHAR*

Given name of the program. Program can be a commercial one a script, or a library or a library component. **@version**: (optional) *NX_CHAR*

Program version plus build number, or commit hash.

@url: (optional) *NX_CHAR*

Description of an ideally ever persistent resource where the source code of the program or this specific compiled version of the program can be found so that the program yields repeatably exactly the same numerical and categorical results.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXprogram/program-field*
- */NXprogram/program@url-attribute*
- */NXprogram/program@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXprogram.nxdl.xml

NXpulser_apm

Status:

base class, extends *NXobject*

Description:

Metadata for laser- and/or voltage-pulsing in atom probe microscopy.

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*, *NXfabrication*, *NXsource*, *NXtransformations*

Structure:

pulse_mode: (optional) *NX_CHAR*

How is field evaporation physically triggered.

Any of these values: `laser` | `voltage` | `laser_and_voltage`

pulse_frequency: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_FREQUENCY*}

Frequency with which the high voltage or laser pulser fires.

pulse_fraction: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_DIMENSIONLESS*}

Fraction of the pulse_voltage that is applied in addition to the standing_voltage at peak voltage of a pulse.

If a standing voltage is applied, this gives nominal pulse fraction (as a function of standing voltage). Otherwise this field should not be present.

pulsed_voltage: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_VOLTAGE*}

In laser pulsing mode the values will be zero so the this field is recommended. However, for voltage pulsing mode it is highly recommended that users report the pulsed_voltage.

pulse_number: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*}

Absolute number of pulses starting from the beginning of the experiment.

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. The standing voltage applied to the sample, relative to system ground.

FABRICATION: (optional) *NXfabrication*

SOURCE: (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 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*} *<=*

Nominal 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.

FABRICATION: (optional) *NXfabrication*

BEAM: (optional) *NXbeam*

Details about specific positions along the focused laser beam which illuminates the (atom probe) specimen. **incidence_vector**: (optional) *NXcollection*

Track time-dependent settings over the course of the measurement how the laser beam in tip space/reconstruction space laser impinges on the sample, i.e. the mean vector is parallel to the laser propagation direction.

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.

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.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXpulser_apm/FABRICATION-group*
- */NXpulser_apm/pulse_fraction-field*
- */NXpulser_apm/pulse_frequency-field*
- */NXpulser_apm/pulse_mode-field*
- */NXpulser_apm/pulse_number-field*

- */NXpulser_apm/pulsed_voltage-field*
- */NXpulser_apm/SOURCE-group*
- */NXpulser_apm/SOURCE/BEAM-group*
- */NXpulser_apm/SOURCE/BEAM/incidence_vector-group*
- */NXpulser_apm/SOURCE/BEAM/pinhole_position-group*
- */NXpulser_apm/SOURCE/BEAM/spot_position-group*
- */NXpulser_apm/SOURCE/FABRICATION-group*
- */NXpulser_apm/SOURCE/name-field*
- */NXpulser_apm/SOURCE/power-field*
- */NXpulser_apm/SOURCE/pulse_energy-field*
- */NXpulser_apm/SOURCE/TRANSFORMATIONS-group*
- */NXpulser_apm/SOURCE/wavelength-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:
NXfabrication

Structure:

design: (optional) *NX_CHAR*

Principle type of the pump.

Any of these values:

- `membrane`
- `rotary_vane`
- `roots`
- `turbo_molecular`

FABRICATION: (optional) *NXfabrication*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXpump/design-field*
- */NXpump/FABRICATION-group*

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) *NX_CHAR*

Path to an *NXtransformations* that defining the axis on which the orientation of the surface depends.

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*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXquadric.nxdl.xml

NXquadrupole_magnet

Status:

base class, extends *NXobject*

Description:

definition for a quadrupole magnet.

Symbols:

No symbol table

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.

- */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. For atom probe the reflectron can be considered an energy_compensation device, which can e.g. be realized technically as via a Poschenrieder lens (see US patent 3863068 or US patents for the reflectron 6740872, or the curved reflectron 8134119 design).

Symbols:

No symbol table

Groups cited:
NXfabrication, *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.

FABRICATION: (optional) *NXfabrication*

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/FABRICATION-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** $\equiv$ **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:

```
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:

```
detector:NXdetector/
        data[128,128,4096]
        region:NXregion/
                @region_type  = "rectangular"
                parent = "data"
                start  = [2]
                count  = [20]
                stride = [32]
                block  = [16]
                downsampled:NXdata/
                        @signal = "maximum"
                        @auxiliary_signals = "copy"
                        maximum[128,128,20]
                        copy[128,128,320]
```

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,

$$(D_0, ..., D_{\mathbf{O}-1}, d_0, ..., d_{\mathbf{R}-1})$$

could be split into a left set of **O** outer dimensions, $\boldsymbol{D}$, and a right set of **R** region dimensions, $\boldsymbol{d}$, 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 $\boldsymbol{R}$ and dimensions $\boldsymbol{d}$. 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 $\mathbf{block}[i] < \mathbf{stride}[i]$ then the downsampling blocks have gaps between them; when `block` matches `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 $(D_0, ..., D_{\mathbf{O}-1}, \mathbf{block}[0] * \mathbf{count}[0], ..., \mathbf{block}[\mathbf{R}-1] * \mathbf{count}[\mathbf{R}-1])$, otherwise the expected shape is $(D_0, ..., D_{\mathbf{O}-1}, \mathbf{count}[0], ..., \mathbf{count}[\mathbf{R}-1])$.

The following figure shows how blocks are used in downsampling:

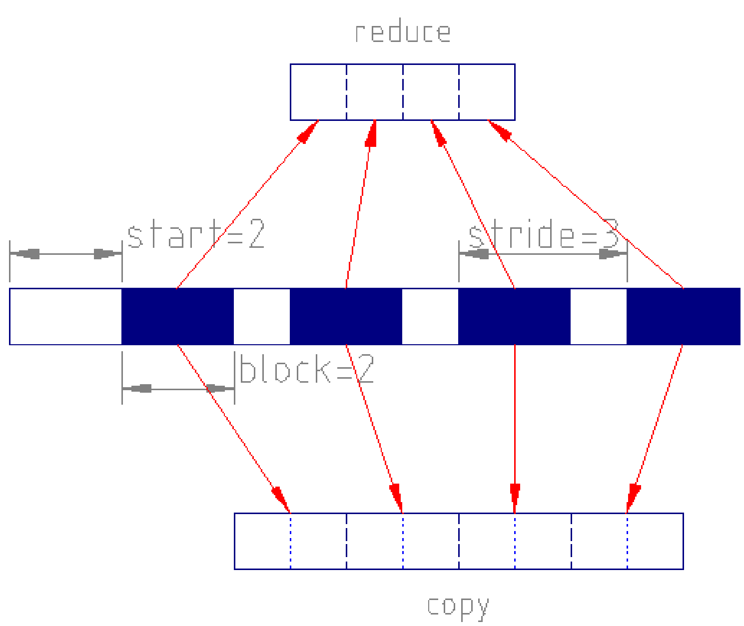


Fig. 14: A selection with $\mathbf{start} = 2, \mathbf{count} = 4, \mathbf{stride} = 3, \mathbf{block} = 2$ from a dataset with shape [13] will result in the `reduce` dataset of shape [4] and a `copy` dataset of shape [8].

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 $\boldsymbol{D}$.

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:
NXfabrication

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*

FABRICATION: (optional) *NXfabrication*

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/FABRICATION-group*
- */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

NXsensor_scan

Status:

application definition, extends *NXobject*

Description:

Application definition for a generic scan using sensors.

In this application definition, times should be specified always together with an UTC offset.

Symbols:

Variables used to set a common size for collected sensor data.

N_scanpoints: The number of scan points measured in this scan.

Groups cited:
NXdata, *NXentry*, *NXenvironment*, *NXinstrument*, *NXpid*, *NXprocess*, *NXsample*, *NXsensor*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR* <=

Obligatory value: `NXsensor_scan`

@version: (required) *NX_CHAR* <=

experiment_identifier: (recommended) *NX_CHAR* <=

experiment_description: (required) *NX_CHAR* <=

start_time: (recommended) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

PROCESS: (required) *NXprocess* <=

Define the program that was used to generate the results file(s) with measured data and metadata. **program**: (required) *NX_CHAR* <=

Commercial or otherwise defined given name of the program (or a link to the instrument software). **@version**: (required) *NX_CHAR*

Either version with build number, commit hash, or description of an (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.

@program_url: (required) *NX_CHAR*

Website of the software.

USER: (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: (recommended) *NX_CHAR* <=

Name of the affiliation of the user at the point in time when the experiment was performed.

address: (recommended) *NX_CHAR* <=

Full address (street, street number, ZIP, city, country) of the user's affiliation.

email: (recommended) *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* <=

ENVIRONMENT: (required) *NXenvironment*

Describes an environment setup for the experiment.

All the setting values of the independently scanned controllers are listed under corresponding NXsensor groups. Similarly, seperate NXsensor groups are used to store the readings from each measurement sensor.

For example, in a temperature-dependent IV measurement, the temperature and voltage must be present as independently scanned controllers and the current sensor must also be present with its readings. **independent_controllers**: (required) *NX_CHAR*

A list of names of NXsensor groups used as independently scanned controllers.

measurement_sensors: (required) *NX_CHAR*

A list of names of NXsensor groups used as measurement sensors.

SENSOR: (required) *NXsensor* <=

value: (required) *NX_FLOAT* (Rank: 1, Dimensions: [N_scanpoints]) {units=*NX_ANY*} <=

For each point in the scan space, either the nominal setpoint of an independently scanned controller or a representative average value of a measurement sensor is registered.

The length of each sensor's data value array stored in this group should be equal to the number of scan points probed in this scan. For every scan point [N], the corresponding sensor value can be picked from index [N]. This allows the scan to be made in any order as the user describes above in the experiment. We get matching values simply using the index of the scan point.

value_timestamp: (recommended) *NX_DATE_TIME*

Timestamp for when the values provided in the value field were registered.

Individual readings can be stored with their timestamps under value_log. This is to timestamp the nominal setpoint or average reading values listed above in the value field.

run_control: (recommended) *NX_CHAR*

@description: (required) *NX_CHAR*

Free-text describing the data acquisition control: an internal sweep using the built-in functionality of the controller device, or a set/wait/read/repeat mechanism.

calibration_time: (required) *NX_DATE_TIME*

ISO8601 datum when calibration was last performed before this measurement. UTC offset should be specified.

DATA: (recommended) *NXdata*

Plot of measured signal as a function of the timestamp of when they have been acquired is also possible.

PID: (required) *NXpid*

SAMPLE: (required) *NXsample* <=

name: (required) *NX_CHAR* <=

DATA: (required) *NXdata* <=

A scan specific representation of the measured signals as a function of the independently controlled environment settings. Plot of every measured signal as a function of the timestamp of when they have been acquired is also possible.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXsensor_scan/ENTRY-group*
- */NXsensor_scan/ENTRY/DATA-group*
- */NXsensor_scan/ENTRY/definition-field*
- */NXsensor_scan/ENTRY/definition@version-attribute*
- */NXsensor_scan/ENTRY/end_time-field*
- */NXsensor_scan/ENTRY/experiment_description-field*
- */NXsensor_scan/ENTRY/experiment_identifier-field*
- */NXsensor_scan/ENTRY/INSTRUMENT-group*
- */NXsensor_scan/ENTRY/INSTRUMENT/ENVIRONMENT-group*
- */NXsensor_scan/ENTRY/INSTRUMENT/ENVIRONMENT/independent_controllers-field*
- */NXsensor_scan/ENTRY/INSTRUMENT/ENVIRONMENT/measurement_sensors-field*
- */NXsensor_scan/ENTRY/INSTRUMENT/ENVIRONMENT/PID-group*
- */NXsensor_scan/ENTRY/INSTRUMENT/ENVIRONMENT/SENSOR-group*
- */NXsensor_scan/ENTRY/INSTRUMENT/ENVIRONMENT/SENSOR/calibration_time-field*
- */NXsensor_scan/ENTRY/INSTRUMENT/ENVIRONMENT/SENSOR/DATA-group*
- */NXsensor_scan/ENTRY/INSTRUMENT/ENVIRONMENT/SENSOR/run_control-field*
- */NXsensor_scan/ENTRY/INSTRUMENT/ENVIRONMENT/SENSOR/run_control@description-attribute*
- */NXsensor_scan/ENTRY/INSTRUMENT/ENVIRONMENT/SENSOR/value-field*
- */NXsensor_scan/ENTRY/INSTRUMENT/ENVIRONMENT/SENSOR/value_timestamp-field*
- */NXsensor_scan/ENTRY/PROCESS-group*
- */NXsensor_scan/ENTRY/PROCESS/program-field*
- */NXsensor_scan/ENTRY/PROCESS/program@program_url-attribute*
- */NXsensor_scan/ENTRY/PROCESS/program@version-attribute*
- */NXsensor_scan/ENTRY/SAMPLE-group*
- */NXsensor_scan/ENTRY/SAMPLE/name-field*
- */NXsensor_scan/ENTRY/start_time-field*
- */NXsensor_scan/ENTRY/USER-group*
- */NXsensor_scan/ENTRY/USER/address-field*
- */NXsensor_scan/ENTRY/USER/affiliation-field*
- */NXsensor_scan/ENTRY/USER/email-field*
- */NXsensor_scan/ENTRY/USER/name-field*
- */NXsensor_scan/ENTRY/USER/orcid-field*
- */NXsensor_scan/ENTRY/USER/telephone_number-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsensor_scan.nxdl.xml

NXseparator

Status:

base class, extends *NXobject*

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) *NX_CHAR* {units=*NX_VOLTAGE*}

read_Efield_current: (optional) *NXlog*

current read from HT supply.

value: (optional) *NX_CHAR* {units=*NX_CURRENT*}

read_Efield_voltage: (optional) *NXlog*

voltage read from HT 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.

- */NXseparator/beamline_distance-field*
- */NXseparator/description-field*
- */NXseparator/read_Bfield_current-group*
- */NXseparator/read_Bfield_current/value-field*
- */NXseparator/read_Bfield_voltage-group*
- */NXseparator/read_Bfield_voltage/value-field*
- */NXseparator/read_Efield_current-group*
- */NXseparator/read_Efield_current/value-field*
- */NXseparator/read_Efield_voltage-group*
- */NXseparator/read_Efield_voltage/value-field*
- */NXseparator/set_Bfield_current-field*
- */NXseparator/set_Efield_voltage-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXseparator.nxdl.xml

NXsimilarity_grouping

Status:

base class, extends *NXobject*

Description:

Metadata to the results of a similarity grouping analysis.

Similarity grouping analyses can be supervised segmentation or machine learning clustering algorithms. These are routine methods which partition the member of a set of objects/geometric primitives into (sub-)groups, features of different type. A plethora of algorithms have been proposed which can be applied also on geometric primitives like points, triangles, or (abstract) features aka objects (including categorical sub-groups).

This base class considers metadata and results of one similarity grouping analysis applied to a set in which objects are either categorized as noise or belonging to a cluster. As the results of the analysis each similarity group, here called feature aka object can get a number of numerical and/or categorical labels.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

c: Cardinality of the set.

n_lbl_num: Number of numerical labels per object.

n_lbl_cat: Number of categorical labels per object.

n_features: Total number of similarity groups aka features, objects, clusters.

Groups cited:
NXprocess

Structure:

cardinality: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of members in the set which is partitioned into features.

number_of_numeric_labels: (optional) *NX_UINT* {units=*NX_UNITLESS*}

How many numerical labels does each feature have.

number_of_categorical_labels: (optional) *NX_UINT* {units=*NX_UNITLESS*}

How many categorical labels does each feature have.

identifier_offset: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_lbl_num]) {units=*NX_UNITLESS*}

Which identifier is the first to be used to label a cluster.

The value should be chosen in such a way that special values can be resolved: * identifier_offset-1 indicates an object belongs to no cluster. * identifier_offset-2 indicates an object belongs to the noise category. Setting for instance identifier_offset to 1 recovers the commonly used case that objects of the noise category get values to -1 and unassigned points to 0. Numerical identifier have to be strictly increasing.

numerical_label: (optional) *NX_UINT* (Rank: 2, Dimensions: [c, n_lbl_num]) {units=*NX_UNITLESS*}

Matrix of numerical label for each member in the set. For classical clustering algorithms this can for instance encode the cluster_identifier.

categorical_label: (optional) *NX_CHAR* (Rank: 2, Dimensions: [c, n_lbl_cat])

Matrix of categorical attribute data for each member in the set.

statistics: (optional) *NXprocess*

In addition to the detailed storage which members was grouped to which feature/group summary statistics are stored under this group. **number_of_unassigned_members**: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_lbl_num]) {units=*NX_UNITLESS*}

Total number of members in the set which are categorized as unassigned.

noise: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_lbl_num]) {units=*NX_UNITLESS*}

Total number of members in the set which are categorized as noise.

number_of_features: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Total number of clusters (excluding noise and unassigned).

feature_identifier: (optional) *NX_UINT* (Rank: 2, Dimensions: [n_features, n_lbl_num]) {units=*NX_UNITLESS*}

Array of numerical identifier of each feature (cluster).

feature_member_count: (optional) *NX_UINT* (Rank: 2, Dimensions: [n_features, n_lbl_num]) {units=*NX_UNITLESS*}

Array of number of members for each feature.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXsimilarity_grouping/cardinality-field*
- */NXsimilarity_grouping/categorical_label-field*
- */NXsimilarity_grouping/identifier_offset-field*
- */NXsimilarity_grouping/number_of_categorical_labels-field*
- */NXsimilarity_grouping/number_of_numeric_labels-field*
- */NXsimilarity_grouping/numerical_label-field*
- */NXsimilarity_grouping/statistics-group*
- */NXsimilarity_grouping/statistics/feature_identifier-field*
- */NXsimilarity_grouping/statistics/feature_member_count-field*
- */NXsimilarity_grouping/statistics/noise-field*
- */NXsimilarity_grouping/statistics/number_of_features-field*
- */NXsimilarity_grouping/statistics/number_of_unassigned_members-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsimilarity_grouping.nxdl.xml

NXslip_system_set

Status:

base class, extends *NXobject*

Description:

Base class for describing a set of crystallographic slip systems.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n: Number of slip systems.

Groups cited:
none

Structure:

lattice_type: (optional) *NX_CHAR*

Any of these values:

- `triclinic`
- `monoclinic`
- `orthorhombic`
- `tetragonal`
- `trigonal`

- `hexagonal`
- `cubic`

miller_plane: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n, i]) {units=*NX_UNITLESS*}

Array of Miller indices which describe the crystallographic plane.

miller_direction: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n, i]) {units=*NX_UNITLESS*}

Array of Miller indices which describe the crystallographic direction.

is_specific: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

For each slip system a marker whether the specified Miller indices refer to the specific slip system or the set of crystallographic equivalent slip systems of the respective family of slip systems.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXslip_system_set/is_specific-field*
- */NXslip_system_set/lattice_type-field*
- */NXslip_system_set/miller_direction-field*
- */NXslip_system_set/miller_plane-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXslip_system_set.nxdl.xml

NXsnsevent

Status:

application definition, extends *NXobject*

Description:

This is a definition for event data from Spallation Neutron Source (SNS) at ORNL.

Symbols:

No symbol table

Groups cited:
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])

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 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) *NXtranslation* <=

distance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

DISK_CHOPPER: (optional) *NXdisk_chopper* <=

distance: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

moderator: (required) *NXmoderator* <=

- **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) *NXaperture* <=

- **x_pixel_offset**: (required) *NX_FLOAT* {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) *NXtranslation* <=
    - **distance**: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

ATTENUATOR: (optional) *NXattenuator* <=

- **distance**: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

POLARIZER: (optional) *NXpolarizer* <=

CRYSTAL: (optional) *NXcrystal* <=

- **type**: (required) *NX_CHAR* <=
- **wavelength**: (required) *NX_FLOAT* {units=*NX_WAVELENGTH*} <=
- **origin**: (required) *NXgeometry* <=
  - **description**: (required) *NX_CHAR* <=
  - **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* {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/instrument/beamline-field*
- */NXsnsevent/ENTRY/instrument/CRYSTAL-group*
- */NXsnsevent/ENTRY/instrument/CRYSTAL/origin-group*
- */NXsnsevent/ENTRY/instrument/CRYSTAL/origin/description-field*
- */NXsnsevent/ENTRY/instrument/CRYSTAL/origin/orientation-group*
- */NXsnsevent/ENTRY/instrument/CRYSTAL/origin/orientation/value-field*
- */NXsnsevent/ENTRY/instrument/CRYSTAL/origin/shape-group*
- */NXsnsevent/ENTRY/instrument/CRYSTAL/origin/shape/description-field*
- */NXsnsevent/ENTRY/instrument/CRYSTAL/origin/shape/shape-field*
- */NXsnsevent/ENTRY/instrument/CRYSTAL/origin/shape/size-field*
- */NXsnsevent/ENTRY/instrument/CRYSTAL/origin/translation-group*
- */NXsnsevent/ENTRY/instrument/CRYSTAL/origin/translation/distance-field*
- */NXsnsevent/ENTRY/instrument/CRYSTAL/type-field*
- */NXsnsevent/ENTRY/instrument/CRYSTAL/wavelength-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR-group*
- */NXsnsevent/ENTRY/instrument/DETECTOR/azimuthal_angle-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/data_x_y-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/distance-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/event_index-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/event_pixel_id-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/event_time_of_flight-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/origin-group*
- */NXsnsevent/ENTRY/instrument/DETECTOR/origin/orientation-group*
- */NXsnsevent/ENTRY/instrument/DETECTOR/origin/orientation/value-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/origin/shape-group*
- */NXsnsevent/ENTRY/instrument/DETECTOR/origin/shape/description-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/origin/shape/shape-field*
- */NXsnsevent/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/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* {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: (required) *NX_UINT* (Rank: 3, Dimensions: [numx, numy, numtof])

data_x_time_of_flight: (required) *NX_UINT* (Rank: 2, Dimensions: [numx, numtof])

data_x_y: (required) *NX_UINT* (Rank: 2, Dimensions: [numx, numy])

data_y_time_of_flight: (required) *NX_UINT* (Rank: 2, Dimensions: [numy, numtof])

distance: (required) *NX_FLOAT* (Rank: 2, Dimensions: [numx, numy]) {units=*NX_LENGTH*} <=

pixel_id: (required) *NX_UINT* (Rank: 2, Dimensions: [numx, numy])

polar_angle: (required) *NX_FLOAT* (Rank: 2, Dimensions: [numx, numy]) {units=*NX_ANGLE*} <=

time_of_flight: (required) *NX_FLOAT* (Rank: 1, Dimensions: [numtof + 1]) {units=*NX_TIME_OF_FLIGHT*} <=

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) *NXtranslation* <=

distance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

DISK_CHOPPER: (optional) *NXdisk_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) *NXfermi_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) *NXmoderator* <=

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) *NXaperture* <=

x_pixel_offset: (required) *NX_FLOAT* {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) *NXtranslation* <=

distance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

ATTENUATOR: (optional) *NXattenuator* <=

distance: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

POLARIZER: (optional) *NXpolarizer* <=

CRYSTAL: (optional) *NXcrystal* <=

type: (required) *NX_CHAR* <=

wavelength: (required) *NX_FLOAT* {units=*NX_WAVELENGTH*} <=

origin: (required) *NXgeometry* <=

description: (required) *NX_CHAR* <=

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* {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])

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.

- */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/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

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

NXspatial_filter

Status:

base class, extends *NXobject*

Description:

Spatial filter to filter entries within a region-of-interest based on their position.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ellipsoids: Number of ellipsoids.

n_hexahedra: Number of hexahedra.

n_cylinders: Number of cylinders.

Groups cited:
NXcg_cylinder_set, *NXcg_ellipsoid_set*, *NXcg_hexahedron_set*, *NXcs_filter_boolean_mask*

Structure:

windowing_method: (optional) *NX_CHAR*

Qualitative statement which specifies which spatial filtering with respective geometric primitives or bitmask is used. These settings are possible:

- entire_dataset, no filter is applied, the entire dataset is used.
- union_of_primitives, a filter with (rotated) geometric primitives. All ions in or on the surface of the primitives are considered while all other ions are ignored.
- bitmasked_points, a boolean array whose bits encode with 1 which ions should be included. Those ions whose bit is set to 0 will be excluded. Users of python can use the bitfield operations of the numpy package to define such bitfields.

Conditions: In the case that windowing_method is entire_dataset all entries are processed. In the case that windowing_method is union_of_primitives, it is possible to specify none or all types of primitives (ellipsoids, cylinder, hexahedra). If no primitives are specified the filter falls back to entire_dataset. In the case that windowing_method is bitmask, the bitmask has to be defined; otherwise the filter falls back to entire dataset.

Any of these values: `entire_dataset` | `union_of_primitives` | `bitmask`

CG_ELLIPSOID_SET: (optional) *NXcg_ellipsoid_set*

CG_CYLINDER_SET: (optional) *NXcg_cylinder_set*

CG_HEXAHEDRON_SET: (optional) *NXcg_hexahedron_set*

CS_FILTER_BOOLEAN_MASK: (optional) *NXcs_filter_boolean_mask*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspatial_filter/CG_CYLINDER_SET-group*
- */NXspatial_filter/CG_ELLIPSOID_SET-group*
- */NXspatial_filter/CG_HEXAHEDRON_SET-group*
- */NXspatial_filter/CS_FILTER_BOOLEAN_MASK-group*
- */NXspatial_filter/windowing_method-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspatial_filter.nxdl.xml

NXspectrum_set

Status:

base class, extends *NXobject*

Description:

Container for reporting a set of spectra.

Symbols:

n_y: Number of pixel in the slow direction.

n_x: Number of pixel in the fast direction.

n_energy: Number of energy bins.

Groups cited:
NXdata, *NXprocess*, *NXprogram*

Structure:

PROCESS: (optional) *NXprocess*

Details how spectra were processed from the detector readings.

source: (optional) *NX_CHAR*

Resolvable data artifact (e.g. filename) from which the all values in the NXdata instances in this NXspectrum_set were loaded during parsing. **@version**: (optional) *NX_CHAR*

An at least as strong as SHA256 hashvalue of the data artifact which source represents digitally to support provenance tracking.

mode: (optional) *NX_CHAR*

Imaging (data collection) mode of the instrument during acquisition of the data in this NXspectrum_set instance.

detector_identifier: (optional) *NX_CHAR*

Link or name of an NXdetector instance with which the data were collected.

PROGRAM: (optional) *NXprogram*

stack: (optional) *NXdata*

Collected spectra for all pixels of a rectangular region-of-interest. This representation supports rectangular scan pattern. **data_counts**: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [n_y, n_x, n_energy]) {units=*NX_UNITLESS*} *<=*

@long_name: (optional) *NX_CHAR* *<=*

Counts

axis_y: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

@long_name: (optional) *NX_CHAR*

Coordinate along y direction

axis_x: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_LENGTH*}

@long_name: (optional) *NX_CHAR*

Coordinate along x direction

axis_energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy]) {units=*NX_ENERGY*}

@long_name: (optional) *NX_CHAR*

Energy

summary: (optional) *NXdata*

Accumulated counts over all pixels of the region-of-interest. This representation supports rectangular scan pattern. **data_counts**: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy]) {units=*NX_UNITLESS*} *<=*

@long_name: (optional) *NX_CHAR* *<=*

Counts

axis_energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy]) {units=*NX_ENERGY*}

@long_name: (optional) *NX_CHAR*

Energy

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspectrum_set/PROCESS-group*
- */NXspectrum_set/PROCESS/detector_identifier-field*
- */NXspectrum_set/PROCESS/mode-field*
- */NXspectrum_set/PROCESS/PROGRAM-group*
- */NXspectrum_set/PROCESS/source-field*
- */NXspectrum_set/PROCESS/source@version-attribute*
- */NXspectrum_set/stack-group*
- */NXspectrum_set/stack/axis_energy-field*
- */NXspectrum_set/stack/axis_energy@long_name-attribute*

- */NXspectrum_set/stack/axis_x-field*
- */NXspectrum_set/stack/axis_x@long_name-attribute*
- */NXspectrum_set/stack/axis_y-field*
- */NXspectrum_set/stack/axis_y@long_name-attribute*
- */NXspectrum_set/stack/data_counts-field*
- */NXspectrum_set/stack/data_counts@long_name-attribute*
- */NXspectrum_set/summary-group*
- */NXspectrum_set/summary/axis_energy-field*
- */NXspectrum_set/summary/axis_energy@long_name-attribute*
- */NXspectrum_set/summary/data_counts-field*
- */NXspectrum_set/summary/data_counts@long_name-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspectrum_set.nxdl.xml

NXspectrum_set_em_eels

Status:

base class, extends *NXobject*

Description:

Container for reporting a set of electron energy loss (EELS) 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 for which the settings, binning, and energy resolution is the same for each scan point.

Symbols:

n_y: Number of pixel per EELS mapping in the slow direction.

n_x: Number of pixel per EELS mapping in the fast direction.

n_energy_loss: Number of electron energy loss bins.

Groups cited:
NXdata, *NXprocess*

Structure:

PROCESS: (optional) *NXprocess*

Details how EELS spectra were processed from the detector readings.

source: (optional) *NX_CHAR*

Typically the name of the input, (vendor) file from which all the NXdata instances in this NXspectrum_set_em_eels were loaded during parsing to represent them in e.g. databases. **@version**: (optional) *NX_CHAR*

An at least as strong as SHA256 hashvalue of the dataset/file which represents the source digitally to support provenance tracking.

program: (optional) *NX_CHAR* <=

Commercial or otherwise given name to the program which was used to process detector data into the EELS spectra stack and summary. **@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.

stack: (optional) *NXdata*

Collected EELS spectra for all pixels of a rectangular region-of-interest. This representation supports rectangular scan pattern. **data_counts**: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [n_y, n_x, n_energy_loss]) {units=*NX_UNITLESS*} <=

Counts for one spectrum per each pixel.

@long_name: (optional) *NX_CHAR* <=

EELS counts

axis_y: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

Pixel center of mass position y-coordinates.

@long_name: (optional) *NX_CHAR*

Coordinate along y direction.

axis_x: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_LENGTH*}

Pixel center of mass position x-coordinates.

@long_name: (optional) *NX_CHAR*

Coordinate along x direction.

axis_energy_loss: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy_loss]) {units=*NX_ENERGY*}

Pixel center of mass energy loss bins.

@long_name: (optional) *NX_CHAR*

Coordinate along energy loss axis.

summary: (optional) *NXdata*

Accumulated EELS spectrum over all pixels of a rectangular region-of-interest. This representation supports rectangular scan pattern. **data_counts**: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy_loss]) {units=*NX_UNITLESS*} <=

Counts for specific energy losses.

@long_name: (optional) *NX_CHAR* <=

Counts electrons with an energy loss within binned range.

axis_energy_loss: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy_loss]) {units=*NX_ENERGY*}

Pixel center of mass energy loss bins.

@long_name: (optional) *NX_CHAR*

Coordinate along energy loss axis.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspectrum_set_em_eels/PROCESS-group*
- */NXspectrum_set_em_eels/PROCESS/program-field*
- */NXspectrum_set_em_eels/PROCESS/program@version-attribute*
- */NXspectrum_set_em_eels/PROCESS/source-field*
- */NXspectrum_set_em_eels/PROCESS/source@version-attribute*
- */NXspectrum_set_em_eels/stack-group*
- */NXspectrum_set_em_eels/stack/axis_energy_loss-field*
- */NXspectrum_set_em_eels/stack/axis_energy_loss@long_name-attribute*
- */NXspectrum_set_em_eels/stack/axis_x-field*
- */NXspectrum_set_em_eels/stack/axis_x@long_name-attribute*
- */NXspectrum_set_em_eels/stack/axis_y-field*
- */NXspectrum_set_em_eels/stack/axis_y@long_name-attribute*
- */NXspectrum_set_em_eels/stack/data_counts-field*
- */NXspectrum_set_em_eels/stack/data_counts@long_name-attribute*
- */NXspectrum_set_em_eels/summary-group*
- */NXspectrum_set_em_eels/summary/axis_energy_loss-field*
- */NXspectrum_set_em_eels/summary/axis_energy_loss@long_name-attribute*
- */NXspectrum_set_em_eels/summary/data_counts-field*
- */NXspectrum_set_em_eels/summary/data_counts@long_name-attribute*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspectrum_set_em_eels.nxdl.xml

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 for which the settings, binning, and energy resolution is the same for each scan point.

IUPAC instead of Siegbahn notation should be used.

Symbols:

n_y: Number of pixel per X-ray mapping in the slow direction

n_x: Number of pixel per X-ray mapping in the fast direction

n_photon_energy: Number of X-ray photon energy (bins)

n_elements: Number of identified elements

n_peaks: Number of peaks

Groups cited:
NXdata, *NXion*, *NXpeak*, *NXprocess*

Structure:

PROCESS: (optional) *NXprocess*

Details how X-ray spectra were processed from the detector readings.

source: (optional) *NX_CHAR*

Typically the name of the input, (vendor) file from which all the NXdata instances in this NXspectrum_set_em_xray were loaded during parsing to represent them in e.g. databases. **@version**: (optional) *NX_CHAR*

An at least as strong as SHA256 hashvalue of the dataset/file which represents the source digitally to support provenance tracking.

program: (optional) *NX_CHAR* *<=*

Commercial or otherwise given name to the program which was used to process detector data into the X-ray spectra stack and summary. **@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.

stack: (optional) *NXdata*

Collected X-ray spectra for all pixels of a rectangular region-of-interest. This representation supports rectangular scan pattern. **data_counts**: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [n_y, n_x, n_photon_energy]) {units=*NX_UNITLESS*} *<=*

@long_name: (optional) *NX_CHAR* *<=*

X-ray photon counts

axis_y: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

@long_name: (optional) *NX_CHAR*

Coordinate along y direction.

axis_x: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_LENGTH*}

@long_name: (optional) *NX_CHAR*

Coordinate along x direction.

axis_photon_energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_photon_energy]) {units=*NX_ENERGY*}

@long_name: (optional) *NX_CHAR*

Photon energy.

summary: (optional) *NXdata*

Accumulated X-ray spectrum over all pixels of a rectangular region-of-interest. This representation supports rectangular scan pattern. **data_counts**: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_photon_energy]) {units=*NX_UNITLESS*} *<=*

@long_name: (optional) *NX_CHAR* *<=*

X-ray photon counts

axis_photon_energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_photon_energy]) {units=*NX_ENERGY*}

@long_name: (optional) *NX_CHAR*

Photon energy

indexing: (optional) *NXprocess*

Details about computational steps how peaks were indexed as elements.

program: (optional) *NX_CHAR* *<=*

Given name of the program that was used to perform this computation.

@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.

element_names: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_elements])

List of the names of identified elements.

PEAK: (optional) *NXpeak*

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) *NXion*

iupac_line_names: (optional) *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.

ELEMENTNAME: (optional) *NXprocess*

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 *under the curve(s)* of a set of peaks. **program**: (optional) *NX_CHAR* *<=*

Given name of the program that was used to perform this computation.

@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.

peaks: (optional) *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.

summary: (optional) *NXdata*

Individual element-specific maps. Individual maps should each be a group and be named according to element_names. **data_counts**: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_y, n_x]) {units=*NX_UNITLESS*} *<=*

@long_name: (optional) *NX_CHAR* *<=*

Accumulated photon counts for observed element.

axis_y: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

@long_name: (optional) *NX_CHAR*

Coordinate along y direction.

axis_x: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_LENGTH*}

@long_name: (optional) *NX_CHAR*

Coordinate along x direction.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspectrum_set_em_xray/indexing-group*
- */NXspectrum_set_em_xray/indexing/element_names-field*
- */NXspectrum_set_em_xray/indexing/ELEMENTNAME-group*
- */NXspectrum_set_em_xray/indexing/ELEMENTNAME/name-field*
- */NXspectrum_set_em_xray/indexing/ELEMENTNAME/peaks-field*
- */NXspectrum_set_em_xray/indexing/ELEMENTNAME/program-field*
- */NXspectrum_set_em_xray/indexing/ELEMENTNAME/program@version-attribute*
- */NXspectrum_set_em_xray/indexing/ELEMENTNAME/summary-group*
- */NXspectrum_set_em_xray/indexing/ELEMENTNAME/summary/axis_x-field*
- */NXspectrum_set_em_xray/indexing/ELEMENTNAME/summary/axis_x@long_name-attribute*
- */NXspectrum_set_em_xray/indexing/ELEMENTNAME/summary/axis_y-field*
- */NXspectrum_set_em_xray/indexing/ELEMENTNAME/summary/axis_y@long_name-attribute*
- */NXspectrum_set_em_xray/indexing/ELEMENTNAME/summary/data_counts-field*
- */NXspectrum_set_em_xray/indexing/ELEMENTNAME/summary/data_counts@long_name-attribute*
- */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*
- */NXspectrum_set_em_xray/PROCESS-group*
- */NXspectrum_set_em_xray/PROCESS/program-field*
- */NXspectrum_set_em_xray/PROCESS/program@version-attribute*
- */NXspectrum_set_em_xray/PROCESS/source-field*
- */NXspectrum_set_em_xray/PROCESS/source@version-attribute*
- */NXspectrum_set_em_xray/stack-group*
- */NXspectrum_set_em_xray/stack/axis_photon_energy-field*
- */NXspectrum_set_em_xray/stack/axis_photon_energy@long_name-attribute*
- */NXspectrum_set_em_xray/stack/axis_x-field*
- */NXspectrum_set_em_xray/stack/axis_x@long_name-attribute*
- */NXspectrum_set_em_xray/stack/axis_y-field*
- */NXspectrum_set_em_xray/stack/axis_y@long_name-attribute*
- */NXspectrum_set_em_xray/stack/data_counts-field*
- */NXspectrum_set_em_xray/stack/data_counts@long_name-attribute*
- */NXspectrum_set_em_xray/summary-group*
- */NXspectrum_set_em_xray/summary/axis_photon_energy-field*
- */NXspectrum_set_em_xray/summary/axis_photon_energy@long_name-attribute*
- */NXspectrum_set_em_xray/summary/data_counts-field*
- */NXspectrum_set_em_xray/summary/data_counts@long_name-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspectrum_set_em_xray.nxdl.xml

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) *NX_CHAR* {units=*NX_CURRENT*}

read_Bfield_voltage: (optional) *NXlog*

voltage read from magnet supply.

value: (optional) *NX_CHAR* {units=*NX_VOLTAGE*}

read_Efield_current: (optional) *NXlog*

current read from HT supply.

value: (optional) *NX_CHAR* {units=*NX_CURRENT*}

read_Efield_voltage: (optional) *NXlog*

voltage read from HT 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.

- */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*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspin_rotator.nxdl.xml

NXspindispersion

Status:

base class, extends *NXobject*

Description:

Subclass of NXelectronanalyser to describe the spin filters in photoemission experiments.

Symbols:

No symbol table

Groups cited:
NXdeflector, *NXlens_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) *NXlens_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:
NXfabrication, *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.

FABRICATION: (optional) *NXfabrication*

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/FABRICATION-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

NXsubsampling_filter

Status:

base class, extends *NXobject*

Description:

Settings of a filter to sample entries based on their value.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

Groups cited:
none

Structure:

linear_range_min_incr_max: (optional) *NX_UINT* (Rank: 1, Dimensions: [3]) {units=*NX_UNITLESS*}

Triplet of the minimum, increment, and maximum value which will be included in the analysis. The increment controls which n-th entry to take.

Take as an example a dataset with 100 entries (their indices start at zero) and the filter set to 0, 1, 99. This will process each entry. 0, 2, 99 will take each second entry. 90, 3, 99 will take only each third entry beginning from entry 90 up to 99.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXsubsampling_filter/linear_range_min_incr_max-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsubsampling_filter.nxdl.xml

NXtransmission

Status:

application definition, extends *NXobject*

Description:

Application definition for transmission experiments

Symbols:

Variables used throughout the experiment

N_wavelengths: Number of wavelength points

N_scans: Number of scans

Groups cited:
NXattenuator, *NXdata*, *NXdetector*, *NXentry*, *NXfabrication*, *NXgrating*, *NXinstrument*, *NXmonochromator*, *NXsample*, *NXslit*, *NXsource*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

This application definition

definition: (required) *NX_CHAR <=*

An application definition for transmission.

Obligatory value: `NXtransmission` **@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.

start_time: (required) *NX_DATE_TIME <=*

Start time of the experiment.

experiment_identifier: (required) *NX_CHAR <=*

Unique identifier of the experiment, such as a (globally persistent)
unique identifier.

- The identifier is usually defined by the facility or principle investigator.
- The identifier enables to link experiments to e.g. proposals.

experiment_description: (optional) *NX_CHAR <=*

An optional free-text description of the experiment. However, details of the experiment should be defined in the specific fields of this application definition rather than in this experiment description.

acquisition_program: (optional) *NXfabrication*

@url: (recommended) *NX_CHAR*

Website of the software

model: (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.

identifier: (required) *NX_CHAR <=*

Version number of the program that was used to generate the result file(s) with measured data and metadata.

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* *<=*

Street address of the user's affiliation.

email: (required) *NX_CHAR* *<=*

Email address of the user.

url: (recommended) *NX_CHAR* *<=*

Author ID defined by reasearch id services, e.g. orcid (https://orcid.org/).

telephone_number: (recommended) *NX_CHAR* *<=*

Telephone number of the user.

instrument: (required) *NXinstrument* *<=*

common_beam_depolarizer: (required) *NX_BOOLEAN*

If true, the incident beam is depolarized.

polarizer: (required) *NX_NUMBER* {units=*NX_ANGLE*}

Polarizer value inside the beam path

time_points: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [N_scans]) {units=*NX_TIME*}

An array of relative scan start time points.

measured_data: (required) *NX_NUMBER* (Rank: 2, Dimensions: [N_scans, N_wavelengths])

Resulting data from the measurement. The length of the 2nd dimension is the number of time points. If it has length one the time_points may be empty.

manufacturer: (recommended) *NXfabrication*

Manufacturer of the instrument.

common_beam_mask: (required) *NXslit*

Common beam mask to shape the incident beam

y_gap: (required) *NX_NUMBER* {units=*NX_UNITLESS*} *<=*

The height of the common beam in percentage of the beam

ref_attenuator: (required) *NXattenuator* *<=*

Attenuator in the reference beam

attenuator_transmission: (required) *NX_FLOAT* *<=*

sample_attenuator: (required) *NXattenuator* *<=*

Attenuator in the sample beam

attenuator_transmission: (required) *NX_FLOAT* *<=*

spectrometer: (required) *NXmonochromator* *<=*

wavelength: (required) *NX_NUMBER* (Rank: 1, Dimensions: [N_wavelengths]) {units=*NX_LENGTH*}

Wavelength value(s) used for the measurement. An array of 1 or more elements. Length defines N_wavelenghts

spectral_resolution: (optional) *NX_NUMBER* {units=*NX_WAVENUMBER*}

Overall spectral resolution of this spectrometer. If several gratings are employed the spectral resoultion should rather be specified for each grating inside the NXgrating group of this spectrometer.

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.

blaze_wavelength: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The blaze wavelength of the grating used.

spectral_resolution: (optional) *NX_NUMBER* {units=*NX_WAVENUMBER*}

Overall spectral resolution of the instrument when this grating is used.

wavelength_range: (required) *NX_NUMBER* (Rank: 1, Dimensions: [2]) {units=*NX_LENGTH*}

Wavelength range in which this grating was used

DETECTOR: (required) *NXdetector <=*

wavelength_range: (required) *NX_NUMBER* (Rank: 1, Dimensions: [2]) {units=*NX_LENGTH*}

Wavelength range in which this detector was used

type: (required) *NX_CHAR <=*

Detector type

Any of these values: `PMT` | `PbS` | `InGaAs`

response_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

Response time of the detector

gain: (optional) *NX_NUMBER*

Detector gain

slit: (required) *NXslit*

Slit setting used for measurement with this detector

type: (required) *NX_CHAR*

Any of these values: `fixed` | `servo`

SOURCE: (required) *NXsource <=*

The lamp used for illumination

type: (required) *NX_CHAR <=*

The type of lamp, e.g. halogen, D2 etc.

Any of these values: `halogen` | `D2`

spectrum: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [N_wavelengths])

The spectrum of the lamp used

wavelength_range: (required) *NX_NUMBER* (Rank: 1, Dimensions: [2]) {units=*NX_LENGTH*}

Wavelength range in which the lamp was used

SAMPLE: (required) *NXsample* <=

Properties of the sample measured

name: (required) *NX_CHAR* <=

data: (required) *NXdata* <=

A default view of the data emitted intensity vs. wavelength. From measured_data plot intensity and wavelength. **@axes**: (required) *NX_CHAR* <=

We recommend to use wavelength as a default attribute, but it can be replaced by any suitable parameter along the X-axis.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXtransmission/ENTRY-group*
- */NXtransmission/ENTRY/acquisition_program-group*
- */NXtransmission/ENTRY/acquisition_program/identifier-field*
- */NXtransmission/ENTRY/acquisition_program/model-field*
- */NXtransmission/ENTRY/acquisition_program@url-attribute*
- */NXtransmission/ENTRY/data-group*
- */NXtransmission/ENTRY/data@axes-attribute*
- */NXtransmission/ENTRY/definition-field*
- */NXtransmission/ENTRY/definition@url-attribute*
- */NXtransmission/ENTRY/definition@version-attribute*
- */NXtransmission/ENTRY/experiment_description-field*
- */NXtransmission/ENTRY/experiment_identifier-field*
- */NXtransmission/ENTRY/instrument-group*
- */NXtransmission/ENTRY/instrument/common_beam_depolarizer-field*
- */NXtransmission/ENTRY/instrument/common_beam_mask-group*
- */NXtransmission/ENTRY/instrument/common_beam_mask/y_gap-field*
- */NXtransmission/ENTRY/instrument/DETECTOR-group*
- */NXtransmission/ENTRY/instrument/DETECTOR/gain-field*

- */NXtransmission/ENTRY/instrument/DETECTOR/response_time-field*
- */NXtransmission/ENTRY/instrument/DETECTOR/slit-group*
- */NXtransmission/ENTRY/instrument/DETECTOR/slit/type-field*
- */NXtransmission/ENTRY/instrument/DETECTOR/type-field*
- */NXtransmission/ENTRY/instrument/DETECTOR/wavelength_range-field*
- */NXtransmission/ENTRY/instrument/manufacturer-group*
- */NXtransmission/ENTRY/instrument/measured_data-field*
- */NXtransmission/ENTRY/instrument/polarizer-field*
- */NXtransmission/ENTRY/instrument/ref_attenuator-group*
- */NXtransmission/ENTRY/instrument/ref_attenuator/attenuator_transmission-field*
- */NXtransmission/ENTRY/instrument/sample_attenuator-group*
- */NXtransmission/ENTRY/instrument/sample_attenuator/attenuator_transmission-field*
- */NXtransmission/ENTRY/instrument/SOURCE-group*
- */NXtransmission/ENTRY/instrument/SOURCE/spectrum-field*
- */NXtransmission/ENTRY/instrument/SOURCE/type-field*
- */NXtransmission/ENTRY/instrument/SOURCE/wavelength_range-field*
- */NXtransmission/ENTRY/instrument/spectrometer-group*
- */NXtransmission/ENTRY/instrument/spectrometer/GRATING-group*
- */NXtransmission/ENTRY/instrument/spectrometer/GRATING/angular_dispersion-field*
- */NXtransmission/ENTRY/instrument/spectrometer/GRATING/blaze_wavelength-field*
- */NXtransmission/ENTRY/instrument/spectrometer/GRATING/spectral_resolution-field*
- */NXtransmission/ENTRY/instrument/spectrometer/GRATING/wavelength_range-field*
- */NXtransmission/ENTRY/instrument/spectrometer/spectral_resolution-field*
- */NXtransmission/ENTRY/instrument/spectrometer/wavelength-field*
- */NXtransmission/ENTRY/instrument/time_points-field*
- */NXtransmission/ENTRY/operator-group*
- */NXtransmission/ENTRY/operator/address-field*
- */NXtransmission/ENTRY/operator/affiliation-field*
- */NXtransmission/ENTRY/operator/email-field*
- */NXtransmission/ENTRY/operator/name-field*
- */NXtransmission/ENTRY/operator/telephone_number-field*
- */NXtransmission/ENTRY/operator/url-field*
- */NXtransmission/ENTRY/SAMPLE-group*
- */NXtransmission/ENTRY/SAMPLE/name-field*
- */NXtransmission/ENTRY/start_time-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXtransmission.nxdl.xml

NXwaveplate

Status:

base class, extends *NXobject*

Description:

A waveplate or retarder.

Symbols:

N_spectrum: Size of the wavelength array for which the refractive index of the material and/or coating is given.

N_wavelengths: Number of discrete wavelengths for which the waveplate is designed. If it operates for a range of wavelengths then N_wavelengths = 2 and the minimum and maximum values of the range should be provided.

Groups cited:
NXsample

Structure:

type: (optional) *NX_CHAR*

Type of waveplate (e.g. achromatic waveplate or zero-order waveplate).

Any of these values:

- `zero-order waveplate`
- `achromatic waveplate`
- `multiple-order waveplate`
- `dual-wavelength waveplate`
- `other`

other_type: (optional) *NX_CHAR*

If you selected 'other' in type describe what it is.

retardance: (optional) *NX_CHAR*

Specify the retardance of the waveplate (e.g. full-wave, half-wave (lambda/2), quarter-wave (lambda/4) plate).

Any of these values:

- `full-wave plate`
- `half-wave plate`
- `quarter-wave plate`

wavelengths: (recommended) *NX_NUMBER* (Rank: 1, Dimensions: [N_wavelengths])

Discrete wavelengths for which the waveplate is designed. If the waveplate operates over an entire range of wavelengths, enter the minimum and maximum values of the wavelength range (in this case N_wavelengths = 2).

diameter: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Diameter of the waveplate.

clear_aperture: (optional) *NX_FLOAT* {units=*NX_UNITLESS*}

Clear aperture of the device (e.g. 90% of diameter for a disc or 90% of length/height for square geometry).

reflectance: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

Average reflectance of the waveplate in percentage.

substrate: (optional) *NXsample*

Describe the material of the substrate of the wave plate in substrate/substrate_material and provide its index of refraction in substrate/index_of_refraction_substrate, if known. **substrate_material**: (optional) *NX_CHAR*

Specify the material of the wave plate. If the device has a coating it should be described in coating/coating_material.

substrate_thickness: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Thickness of the wave plate substrate.

index_of_refration_substrate: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [2, N_spectrum]) {units=*NX_UNITLESS*}

Complex index of refraction of the wave plate substrate. Specify at given wavelength (or energy, wavenumber etc.) values.

coating: (optional) *NXsample*

Is the wave plate coated? If yes, specify the type and material of the coating and the wavelength range for which it is designed. If known, you may also provide its index of refraction. **coating_type**: (optional) *NX_CHAR*

Specify the coating type (e.g. dielectric, anti-reflection (AR), multilayer coating etc.).

coating_material: (optional) *NX_CHAR*

Specify the coating material.

coating_thickness: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Thickness of the coating.

wavelength_range_coating: (recommended) *NX_NUMBER* (Rank: 1, Dimensions: [2])

Wavelength range for which the coating is designed. Enter the minimum and maximum values of the wavelength range.

index_of_refraction_coating: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [2, N_spectrum]) {units=*NX_UNITLESS*}

Complex index of refraction of the coating. Specify at given spectral values (wavelength, energy, wavenumber etc.).

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXwaveplate/clear_aperture-field*
- */NXwaveplate/coating-group*
- */NXwaveplate/coating/coating_material-field*
- */NXwaveplate/coating/coating_thickness-field*
- */NXwaveplate/coating/coating_type-field*
- */NXwaveplate/coating/index_of_refraction_coating-field*
- */NXwaveplate/coating/wavelength_range_coating-field*
- */NXwaveplate/diameter-field*
- */NXwaveplate/other_type-field*
- */NXwaveplate/reflectance-field*
- */NXwaveplate/retardance-field*
- */NXwaveplate/substrate-group*
- */NXwaveplate/substrate/index_of_refration_substrate-field*
- */NXwaveplate/substrate/substrate_material-field*
- */NXwaveplate/substrate/substrate_thickness-field*
- */NXwaveplate/type-field*
- */NXwaveplate/wavelengths-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXwaveplate.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

$$g_2(\boldsymbol{Q}, t) = \frac{\langle I(\boldsymbol{Q}, t\prime) I(\boldsymbol{Q}, t\prime + t)\rangle}{\langle I(\boldsymbol{Q}, t\prime)\rangle^2}; t > 0$$

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")

Any of these values: `one_array` | `data_exchange_keys` | `other`

g2_derr: (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 `g2`. **@storage_mode**: (required) *NX_CHAR*

Any of these values: `one_array` | `data_exchange_keys` | `other`

G2_unnormalized: (optional) *NX_NUMBER* {units=*NX_ANY*}

unnormalized intensity auto-correlation function.

Specifically, `g2` without the denominator. The data should be in the same format as `g2`. **@storage_mode**: (required) *NX_CHAR*

Any of these values: `one_array` | `data_exchange_keys` | `other`

delay_difference: (optional) *NX_INT* {units=*NX_COUNT*}

[1] mask: `NXxpcs:/entry/instrument/masks-group`

[2] NeXus 2-D data and axes: https://manual.nexusformat.org/classes/base_classes/NXdata.html#nxdata

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 `g2` 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:

$$C(\boldsymbol{Q}, t_1, t_2) = \frac{\langle I(\boldsymbol{Q}, t_1) I(\boldsymbol{Q}, t_2)\rangle}{\langle I(\boldsymbol{Q}, t_1)\rangle \langle I(\boldsymbol{Q}, t_2)\rangle}$$

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 `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.

@storage_mode: (required) *NX_CHAR*

storage_mode describes the format of the data to be loaded

We encourage the documention of other formats represented here.

Any of these values:

- `one_array_q_first`
- `one_array_q_last`

- `data_exchange_keys`
- `other`

@baseline_reference: (required) *NX_INT*

baseline is the expected value of a full decorrelation

The baseline is a constant value added to the functional form of the auto-correlation function. This value is required.

Any of these values: `0` | `1`

@time_origin_location: (required) *NX_CHAR*

time_origin_location is the location of the origin

Any of these values: `upper_left` | `lower_left`

@populated_elements: (required) *NX_CHAR*

populated_elements describe the elements of the 2D array that are populated with data

Any of these values: `all` | `upper_half` | `lower_half`

g2_from_two_time_corr_func: (optional) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

frame weighted average along the diagonal direction in `two_time_corr_func`

The data format and description should be consistent with that found in "/NXxpcs/entry/data/g2"

- iterable 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`

g2_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`

g2_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 exists in the CXI documentation (See Fig 11 vs Fig 12).

Additionally, not all *NXdetector* dependencies are inherited from AreaDetector or other control systems. `frame_time` is used to convert `delay_difference` to seconds. `frame_time` field could be missing from AreaDetector or may either be *acquire_period* or *acquire_time*, depending on the detector model and the local implementation. **description**: (optional) *NX_CHAR* *<=*

Detector name.

distance: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Distance between sample and detector.

count_time: (required) *NX_NUMBER* {units=*NX_TIME*} *<=*

Exposure time of frames, s.

frame_time: (required) *NX_NUMBER* {units=*NX_TIME*}

Exposure period (time between frame starts) of frames, s

beam_center_x: (required) *NX_NUMBER* {units=*NX_LENGTH*}

Position of beam center, x axis, in detector's coordinates.

beam_center_y: (required) *NX_NUMBER* {units=*NX_LENGTH*}

Position of beam center, y axis, in detector's coordinates.

x_pixel_size: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Length of pixel in x direction.

[3] Coherent X-ray Imaging Data Bank: https://cxidb.org/cxi.html

y_pixel_size: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Length of pixel in y direction.

masks: (optional) *NXnote*

Data masks or mappings to regions of interest (roi) for specific Q values

Fields in this `masks` group describe regions of interest in the data by either a mask to select pixels or to associate a *map* of rois with a (one-dimensional) *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 NXxcps/entry/data and NXxpcs/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 *NXcanSAS* to represent standard SAXS results or development of new NeXus definitions for GI-SAXS or other reciprocal space intensity mapping. **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., $Q(r)$, φ), but preferable use the seperate φ 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 φ 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) *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.

- */NXxpcs/entry-group*
- */NXxpcs/entry/data-group*
- */NXxpcs/entry/data/delay_difference-field*
- */NXxpcs/entry/data/delay_difference@storage_mode-attribute*
- */NXxpcs/entry/data/frame_average-field*
- */NXxpcs/entry/data/frame_sum-field*
- */NXxpcs/entry/data/g2-field*
- */NXxpcs/entry/data/g2@storage_mode-attribute*
- */NXxpcs/entry/data/g2_derr-field*
- */NXxpcs/entry/data/g2_derr@storage_mode-attribute*
- */NXxpcs/entry/data/G2_unnormalized-field*

- */NXxpcs/entry/data/G2_unnormalized@storage_mode-attribute*
- */NXxpcs/entry/definition-field*
- */NXxpcs/entry/end_time-field*
- */NXxpcs/entry/entry_identifier-field*
- */NXxpcs/entry/entry_identifier_uuid-field*
- */NXxpcs/entry/instrument-group*
- */NXxpcs/entry/instrument/DETECTOR-group*
- */NXxpcs/entry/instrument/DETECTOR/beam_center_x-field*
- */NXxpcs/entry/instrument/DETECTOR/beam_center_y-field*
- */NXxpcs/entry/instrument/DETECTOR/count_time-field*
- */NXxpcs/entry/instrument/DETECTOR/description-field*
- */NXxpcs/entry/instrument/DETECTOR/distance-field*
- */NXxpcs/entry/instrument/DETECTOR/frame_time-field*
- */NXxpcs/entry/instrument/DETECTOR/x_pixel_size-field*
- */NXxpcs/entry/instrument/DETECTOR/y_pixel_size-field*
- */NXxpcs/entry/instrument/incident_beam-group*
- */NXxpcs/entry/instrument/incident_beam/extent-field*
- */NXxpcs/entry/instrument/incident_beam/incident_energy-field*
- */NXxpcs/entry/instrument/incident_beam/incident_energy_spread-field*
- */NXxpcs/entry/instrument/incident_beam/incident_polarization_type-field*
- */NXxpcs/entry/instrument/masks-group*
- */NXxpcs/entry/instrument/masks/dynamic_phi_list-field*
- */NXxpcs/entry/instrument/masks/dynamic_q_list-field*
- */NXxpcs/entry/instrument/masks/dynamic_roi_map-field*
- */NXxpcs/entry/instrument/masks/static_q_list-field*
- */NXxpcs/entry/instrument/masks/static_roi_map-field*
- */NXxpcs/entry/NOTE-group*
- */NXxpcs/entry/sample-group*
- */NXxpcs/entry/sample/position_x-group*
- */NXxpcs/entry/sample/position_y-group*
- */NXxpcs/entry/sample/position_z-group*
- */NXxpcs/entry/sample/temperature-field*
- */NXxpcs/entry/sample/temperature_set-field*
- */NXxpcs/entry/scan_number-field*
- */NXxpcs/entry/start_time-field*
- */NXxpcs/entry/twotime-group*

- */NXxpcs/entry/twotime/g2_err_from_two_time_corr_func-field*
- */NXxpcs/entry/twotime/g2_err_from_two_time_corr_func@storage_mode-attribute*
- */NXxpcs/entry/twotime/g2_err_from_two_time_corr_func_partials-field*
- */NXxpcs/entry/twotime/g2_from_two_time_corr_func-field*
- */NXxpcs/entry/twotime/g2_from_two_time_corr_func@baseline_reference-attribute*
- */NXxpcs/entry/twotime/g2_from_two_time_corr_func@first_point_for_fit-attribute*
- */NXxpcs/entry/twotime/g2_from_two_time_corr_func@storage_mode-attribute*
- */NXxpcs/entry/twotime/g2_from_two_time_corr_func_partials-field*
- */NXxpcs/entry/twotime/g2_from_two_time_corr_func_partials@baseline_reference-attribute*
- */NXxpcs/entry/twotime/g2_from_two_time_corr_func_partials@storage_mode-attribute*
- */NXxpcs/entry/twotime/two_time_corr_func-field*
- */NXxpcs/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:

download file	description
`/_static/nxdl_vocabulary.html`	Human-readable HTML list of anchors, by vocabulary term, with links to the manual.
`/_static/nxdl_vocabulary.json`	vocabulary list by key in JSON format[2]
`/_static/nxdl_vocabulary.txt`	list of all anchors, sorted alphabetically
`/_static/nxdl_vocabulary.yml`	vocabulary list by key in YAML format[3]

[2] JSON: https://www.w3schools.com/whatis/whatis_json.asp
[3] YAML https://yaml.org

CHAPTER

FOUR

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
https://manual.nexusformat.org/doxygen/html-c/

C++
https://manual.nexusformat.org/doxygen/html-cpp/ https://github.com/nexusformat/code/tree/master/bindings/cpp

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.

C data type	F90 data type
`int, int`	`integer`
`char*`	`character(len=*)`
`NXhandle, NXhandle*`	`type(NXhandle)`
`NXstatus`	`integer`
`int[]`	`integer(:)`
`void*`	`real(:)` or `integer(:)` or `character(len=*)`
`NXlink a`, `NXlink* a`	`type(NXlink)`

The parameters in the next table, defined in `NXmodule`, may be used in defining variables.

Name	Description	Value
`NX_MAXRANK`	Maximum number of dimensions	`32`
`NX_MAXNAMELEN`	Maximum length of NeXus name	`64`
`NXi1`	Kind parameter for a 1-byte integer	`selected_int_kind(2)`
`NXi2`	Kind parameter for a 2-byte integer	`selected_int_kind(4)`
`NXi4`	Kind parameter for a 4-byte integer	`selected_int_kind(8)`
`NXr4`	Kind parameter for a 4-byte real	`kind(1.0)`
`NXr8`	Kind parameter for an 8-byte real	`kind(1.0D0)`

The bindings are listed at 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

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.

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), 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

```
#!/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

```
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

```
    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

```
int idata[][] = new idata[10][20];
int iDim[] = new int[2];

// put some data into idata.......

// write idata
iDim[0] = 10;
iDim[1] = 20;
nf.makedata("idata",NexusFile.NX_INT32,2,iDim);
nf.opendata("idata");
nf.putdata(idata);

// read idata
nf.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

```
String ame = "Alle meine Entchen";
nf.makedata("string_data",NexusFile.NX_CHAR,
        1,ame.length()+2);
nf.opendata("string_data");
nf.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 \
        --with-idlroot=/path/to/idl/installation \
        --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

name	description
Reading and Writing	
NXUwriteglobals	Writes all the valid global attributes of a file.
NXUwritegroup	Opens a group (creating it if necessary).
NXUwritedata	Opens a data item (creating it if necessary) and writes data and its units.
NXUreaddata	Opens and reads a data item and its units.
NXUwritehistogra	Opens one dimensional data item (creating it if necessary) and writes histogram centers and their units.
NXUreadhistogram	Opens and reads a one dimensional data item and converts it to histogram bin boundaries.
NXUsetcompress	Defines the compression algorithm and minimum dataset size for subsequent write operations.
Finding Groups, Data, and Attributes	
NXUfindclass	Returns the name of a group of the specified class if it is contained within the currently open group.
NXUfinddata	Checks whether a data item of the specified name is contained within the currently open group.
NXUfindattr	Checks whether the currently open data item has the specified attribute.
NXUfindsignal	Searches the currently open group for a data item with the specified SIGNAL attribute.
NXUfindaxis	Searches the currently open group for a data item with the specified AXIS attribute.
Finding Linked Groups	
NXUfindlink	Finds another link to the specified NeXus data item and opens the group it is in.
NXUresumelink	Reopens the original group from which NXUfindlink was used.

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)

CHAPTER

FIVE

NEXUS 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 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/, 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/. 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* or *Application Definitions*) for consideration in the NeXus standard. These community contributions will be entered in the *Contributed Definitions* and will be curated according to procedures set forth by the *NIAC: The NeXus International Advisory Committee*.

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 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 NIAC. 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 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

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

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

NeXus NXDL class definitions
https://github.com/nexusformat/definitions

NeXus GitHub *organization*
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) 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

CHAPTER

SIX

INSTALLATION

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

CHAPTER

SEVEN

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[1] which you may browse and download. There is a cursory analysis[2] of every file in this repository as to whether it can be read as HDF5 or NeXus HDF5. The analysis code[3], 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 (http://www.ncnr.nist.gov/dave/)
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)
The Data Analysis WorkbeNch (DAWN) project is an eclipse based workbench for doing scientific data anaylsis. It offers generic visualisation, and domain specific processing.

GDA (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 (https://archive.ansto.gov.au/ResearchHub/OurInfrastructure/ACNS/Facilities/Computing/GumTree/index.htm)
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)
IDL is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation.

IgorPro (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.

ISAW (ftp://ftp.sns.gov/ISAW/)
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/>)
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/)
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/)
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/)
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)

silx (http://www.silx.org/doc/silx/latest/)
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[4] 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] 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], reads *NXstxm*

7.6.3 Language API: *IgorPro*

- **HDF5gateway**[9] makes it easy to read a HDF5 file (including NeXus) into an IgorPro[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**[11] has java code to read[12] and write[13] HDF5 NeXus files (generic NeXus, not tied to specific application definitions).
- `NXreader.zip`[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://dawnsci.org/
[12] read: https://github.com/DawnScience/scisoft-core/blob/master/uk.ac.diamond.scisoft.analysis/src/uk/ac/diamond/scisoft/analysis/io/NexusHDF5Loader.java
[13] write: https://github.com/DawnScience/dawnsci/blob/master/org.eclipse.dawnsci.hdf5/src/org/eclipse/dawnsci/hdf5/nexus/NexusFileHDF5.java
[14] https://github.com/nexusformat/definitions/files/4107439/NXreader.zip

7.6.5 Language API: *Python*

- **Dials**[15] has python (and some C++) code for reading *NXmx*[16]
  - *cctbx.xfel* code for writing[17] *NXmx* master files for JF16M at SwissFEL
- **h5py**[Page 1116, 18]
  HDF5 for Python (h5py) is a general-purpose Python interface to HDF5.
- **Mantis**[19], with NeXus-specific python code[20], reads *NXstxm*
- **nexusformat**[21] **NeXus package for Python**
  Provides an API to open, create, plot, and manipulate NeXus data.
- **SasView**[22] has python code to read[23] and write[24] *NXcanSAS*

7.6.6 Language API: *mixed*

- **FOCUS**: `focus.zip`[25] contains:
  - An example FOCUS file
  - focusreport: A h5py program which skips through a list of files and prints statistics
  - focusreport.tcl, same as above but in Tcl using the Swig generated binding to NAPI
  - i80.f contains a F77 routine for reading FOCUS files into Ida. The routine is RRT_in_Foc.
- **ZEBRA**: `zebra.zip`[26] contains:
  - an example file
  - zebra-to-ascii, a h5py script which dumps a zebra file to ASCII
  - `TRICSReader.*` for reading ZEBRA files in C++ using C-NAPI calls

15 https://dials.github.io/
16 read: https://github.com/cctbx/dxtbx/blob/master/format/nexus.py
17 write: https://github.com/cctbx/cctbx_project/blob/master/xfel/swissfel/jf16m_cxigeom2nexus.py
18 http://docs.h5py.org
19 Mantis: http://spectromicroscopy.com/
20 python code: https://bitbucket.org/mlerotic/spectromicroscopy/src/default/
21 https://github.com/nexpy/nexusformat
22 https://www.sasview.org/
23 read: https://github.com/SasView/sasview/blob/master/src/sas/sascalc/dataloader/readers/cansas_reader_HDF5.py
24 write: https://github.com/SasView/sasview/blob/master/src/sas/sascalc/file_converter/nxcansas_writer.py
25 https://github.com/nexusformat/definitions/files/4107386/focus.zip
26 https://github.com/nexusformat/definitions/files/4107416/zebra.zip

CHAPTER

EIGHT

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:

- *release v2020.10 <https://github.com/nexusformat/definitions/wiki/releasenotes__v2020.10>* of NeXus Definitions

2020-01:

- *release v2020.1 <https://github.com/nexusformat/definitions/wiki/releasenotes__v2020.1>* of NeXus Definitions

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.Koennecke@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

Publishing Information

This manual built Feb 09, 2024.

See also:

This document is available in these formats online:

HTML
: https://manual.nexusformat.org/

PDF
: https://manual.nexusformat.org/_static/NeXusManual.pdf

A very brief overview (title: *NeXus for the Impatient*) is also available (separate from the manual).

HTML
: https://manual.nexusformat.org/impatient/

PDF
: https://manual.nexusformat.org/_static/NXImpatient.pdf

INDEX

A

B

C

D

E

F

G

H

I

M

N

Q

R

S

T

U

V

W

X