
nexus
Release v2024.02

NIAC, <https://www.nexusformat.org>

May 21, 2025

CONTENTS

1 NeXus: User Manual	3
1.1 NeXus Introduction	3
1.2 NeXus Design	17
1.3 Constructing NeXus Files and Application Definitions	58
1.4 Strategies for storing information in NeXus data files	70
1.5 Verification and validation of files	73
1.6 Frequently Asked Questions	73
2 Examples of writing and reading NeXus data files	77
2.1 Code Examples in Various Languages	77
2.2 Visualization tools	110
2.3 Examples for Specific Instruments	114
2.4 Other tools to handle NeXus data files	130
3 NeXus: Reference Documentation	131
3.1 Introduction to NeXus definitions	131
3.2 NXDL: The NeXus Definition Language	134
3.3 NeXus Class Definitions	160
4 NAPI: NeXus Application Programmer Interface (frozen)	1063
4.1 Status	1063
4.2 Overview	1063
4.3 Core API	1063
4.4 Utility API	1070
4.5 Building Programs	1071
4.6 Reporting Bugs in the NeXus API	1072
5 NeXus Community	1073
5.1 NeXus Webpage	1073
5.2 Contributed Definitions	1073
5.3 Other Ways NeXus Coordinates with the Scientific Community	1073
6 Installation	1077
6.1 Precompiled Binary Installation	1077
6.2 Source Installation	1078
6.3 Releases	1078
7 NeXus Utilities	1081
7.1 Utilities supplied with NeXus	1081
7.2 Validation	1082
7.3 Other Utilities	1082

7.4	Data Analysis	1083
7.5	HDF Tools	1084
7.6	Language APIs for NeXus and HDF5	1085
8	Brief history of NeXus	1087
9	About these docs	1091
9.1	Authors	1091
9.2	Colophon	1091
9.3	Revision History	1092
9.4	Copyright and Licenses	1092
Index		1093



<https://www.nexusformat.org/>

NEXUS: USER MANUAL



1.1 NeXus Introduction

NeXus¹ 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

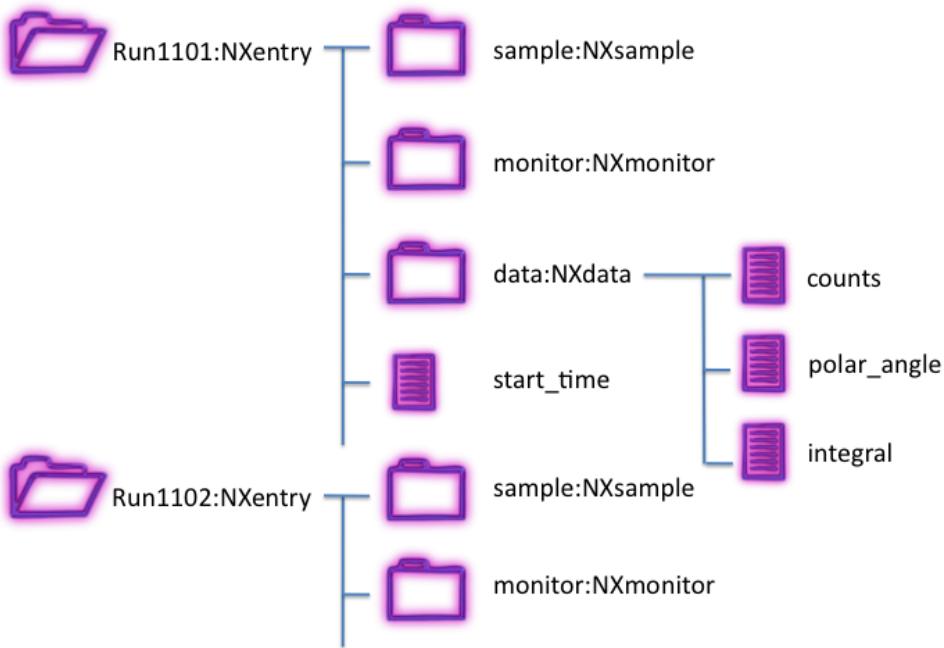
Links are used to represent the same information in different places.

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`

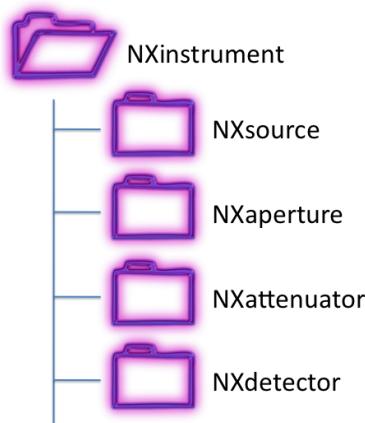
A `NXentry` group will often contain a group with class `NXsample`. This group contains information pertaining to the sample, such as its chemical composition, mass, and environment variables (temperature, pressure, magnetic

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

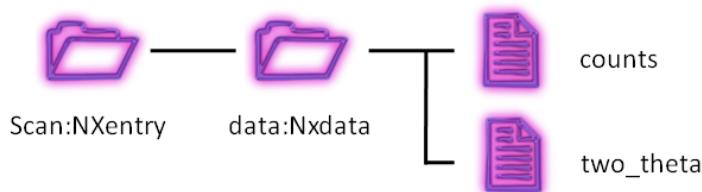


Since an instrument can include several beamline components each defined by several parameters, the components are each specified by a separate group. This hides the complexity from generic file browsers, but makes the information available in an intuitively obvious way if it is required.

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

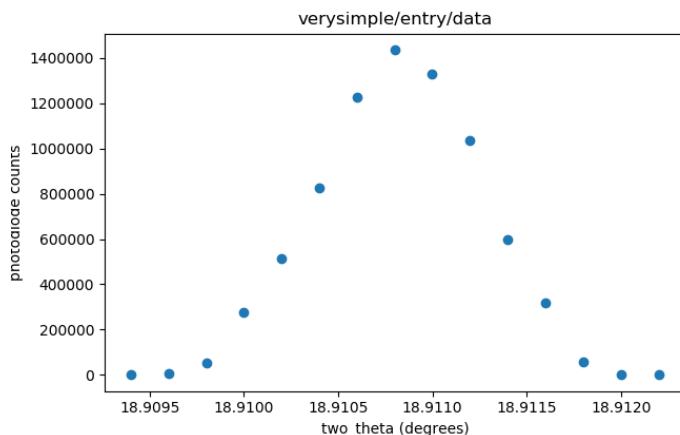
```

1  verysimple.nx5 : NeXus data file
2      @default = "entry"
3      entry:NXentry
4          @NX_class = NXentry
5          @default = "data"
6          data:NXdata
7              @NX_class = NXdata
8              @signal = "counts"
9              @axes = "two_theta"
10             @two_theta_indices = [0]
11             counts:int32[15] = [1193, 4474, 53220, ..., 1000]
12                 @units = "counts"
13                 @long_name = photodiode counts
14             two_theta:float64[15] = [18.9094, 18.9096, ..., 18.9122]
15                 @units = "degrees"
16                 @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

```

1  #!/usr/bin/env python
2  """uses h5py to build the verysimple.nx5 data file"""
3
4  import h5py
5
6  angle = [18.9094, 18.9096, 18.9098, 18.91, 18.9102,
7          18.9104, 18.9106, 18.9108, 18.911, 18.9112,
8          18.9114, 18.9116, 18.9118, 18.912, 18.9122]
9  diode = [1193, 4474, 53220, 274310, 515430, 827880,

```

(continues on next page)

(continued from previous page)

```

10      1227100, 1434640, 1330280, 1037070, 598720,
11      316460, 56677, 1000, 1000]

12
13 with h5py.File('verysimple.nx5', 'w') as f:
14     f.attrs['default'] = 'entry'

15
16     nxentry = f.create_group('entry')
17     nxentry.attrs["NX_class"] = 'NXentry'
18     nxentry.attrs['default'] = 'data'

19
20     nxdata = nxentry.create_group('data')
21     nxdata.attrs["NX_class"] = 'NXdata'
22     nxdata.attrs['signal'] = 'counts'
23     nxdata.attrs['axes'] = 'two_theta'
24     nxdata.attrs['two_theta_indices'] = [0,]

25
26     tth = nxdata.create_dataset('two_theta', data=angle)
27     tth.attrs['units'] = 'degrees'
28     tth.attrs['long_name'] = 'two_theta (degrees)'

29
30     counts = nxdata.create_dataset('counts', data=diode)
31     counts.attrs['units'] = 'counts'
32     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

```

1 <?xml version="1.0" ?>
2 <definition
3   xmlns="http://definition.nexusformat.org/nxdl/3.1"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="http://definition.nexusformat.org/nxdl/3.1 ../nxdl.xsd"
6   category="base"
7   name="NXverysimple"
8   type="group" extends="NXobject">
9
10 <doc>

```

(continues on next page)

(continued from previous page)

```

11 A very simple NeXus NXDL file
12 </doc>
13 <group type="NXentry">
14   <group type="NXdata">
15     <field name="counts" type="NX_INT" units="NX_UNITLESS">
16       <doc>counts recorded by detector</doc>
17     </field>
18     <field name="two_theta" type="NX_FLOAT" units="NX_ANGLE">
19       <doc>rotation angle of detector arm</doc>
20     </field>
21   </group>
22 </group>
23 </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

```

1 NXopendata (fileID, "wavelength");
2 NXgetdata (fileID, lambda);
3 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

```
1 entry:NXentry
2   instrument:NXinstrument
3     detector:NXdetector
4       data:[]
5         @long_name = "strip detector 1-D array"
6       bins:[0, 1, 2, ... 1023]
7         @long_name = "bin index numbers"
8       sample:NXsample
9         name = "zeolite"
10      data:NXdata
11        @signal = "data"
12        @axes = ["bins", "bins"]
13        @bins_indices = [0, 1]
14        data --> /entry/instrument/detector/data
15        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.

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.

¹ `#include "napi.h"`

²

(continues on next page)

¹ <https://github.com/nexusformat/code/releases/>

(continued from previous page)

```

3 int main()
4 {
5     /* we start with known arrays tth and counts, each length n */
6     NXhandle fileID;
7     NXopen ("NXfile.nxs", NXACC_CREATE, &fileID);
8     NXmakegroup (fileID, "Scan", "NXentry");
9     NXopengroup (fileID, "Scan", "NXentry");
10    NXmakegroup (fileID, "data", "NXdata");
11    NXopengroup (fileID, "data", "NXdata");
12        NXmakedata (fileID, "two_theta", NX_FLOAT32, 1, &n);
13        NXopendata (fileID, "two_theta");
14            NXputdata (fileID, tth);
15            NXputattr (fileID, "units", "degrees", 7, NX_CHAR);
16            NXclosedata (fileID); /* two_theta */
17            NXmakedata (fileID, "counts", NX_FLOAT32, 1, &n);
18            NXopendata (fileID, "counts");
19                NXputdata (fileID, counts);
20                NXclosedata (fileID); /* counts */
21                NXclosegroup (fileID); /* data */
22                NXclosegroup (fileID); /* Scan */
23    NXclose (&fileID);
24    return;
25 }
```

program analysis

1. line 7:

Open the file `NXfile.nxs` with *create* access (implying write access). NAPI² returns a file identifier of type `NXhandle`.

2. line 7:

Next, we create the `NXentry` group to contain the scan using `NXmakegroup()` and then open it for access using `NXopengroup()`.³

3. line 10:

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

4. line 12:

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

5. line 14:

Write the data using `NXputdata()`.

6. line 15:

With the field still open, we can also add some field attributes, such as the data units,⁵ which are specified as a character string (`type="NX_CHAR"`)⁶ 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

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*

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*.)

Reading a simple NeXus file using NAPI

```

1 NXopen ('NXfile.nxs', NXACC_READ, &fileID);
2 NX.opengroup (fileID, "Scan", "NXentry");
3 NX.opengroup (fileID, "data", "NXdata");
4 NX.opendata (fileID, "two_theta");
5 NXgetinfo (fileID, &rank, dims, &datatype);
6 NXmalloc ((void **) &tth, rank, dims, datatype);
7 NXgetdata (fileID, tth);
8 NXclosedata (fileID);
9 NXclosegroup (fileID);
10 NXclosegroup (fileID);
11 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`

```

1 %> nxbrowse lrcs3701.nxs
2
3 NXBrowse 3.0.0. Copyright (C) 2000 R. Osborn, M. Koennecke, P. Klosowski
4   NeXus_version = 1.3.3
5   file_name = lrcs3701.nxs
6   file_time = 2001-02-11 00:02:35-0600
7   user = EAG/RO
8 NX> dir
9   NX Group : Histogram1 (NXentry)
10  NX Group : Histogram2 (NXentry)
11 NX> open Histogram1
12 NX/Histogram1> dir
13   NX Data  : title[44] (NX_CHAR)
14   NX Data  : analysis[7] (NX_CHAR)
```

(continues on next page)

(continued from previous page)

```

15 NX Data : start_time[24] (NX_CHAR)
16 NX Data : end_time[24] (NX_CHAR)
17 NX Data : run_number (NX_INT32)
18 NX Group : sample (NXsample)
19 NX Group : LRMECS (NXinstrument)
20 NX Group : monitor1 (NXmonitor)
21 NX Group : monitor2 (NXmonitor)
22 NX Group : data (NXdata)
23 NX/Histogram1> read title
24 title[44] (NX_CHAR) = MgB2 PDOS 43.37g 8K 120meV E0@240Hz T0@120Hz
25 NX/Histogram1> open data
26 NX/Histogram1/data> dir
27 NX Data : title[44] (NX_CHAR)
28 NX Data : data[148,750] (NX_INT32)
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.

⁸ <https://github.com/nexusformat/code/blob/master/applications/NXbrowse/NXbrowse.c>

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.

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`, `hs1a` and `cmyk` are $(n \times m \times 4)$ arrays, where the 4 channels are the colour channels appropriately. If the image data does not contain an alpha channel, then the array should simply be $(n \times m \times 3)$. Allowed values include:

- scalar (0-D data)
- scaler DEPRECATED, use scalar
- spectrum (1-D data)
- image (2-D data)
- rgb-image (3-D data)
- rgba-image (3-D data)
- hsl-image (3-D data)
- hsla-image (3-D data)
- cmyk-image (3-D data)
- vertex (3-D data)

File attributes

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

Links

Python h5py code to make NeXus links

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

- `{hdf5_object}.id.link()`
- `h5py.ExternalLink()`

Links are pointers to existing data somewhere else. The concept is very much like symbolic links in a unix filesystem. The NeXus definition sometimes requires to have access to the same data in different groups in the same file. For

example: detector data is stored in the `NXinstrument/NXdetector` group but may be needed in `NXdata` for automatic plotting. Rather than replicating the data, NeXus uses links in such situations. See the [figure](#) for a more descriptive representation of the concept of linking.

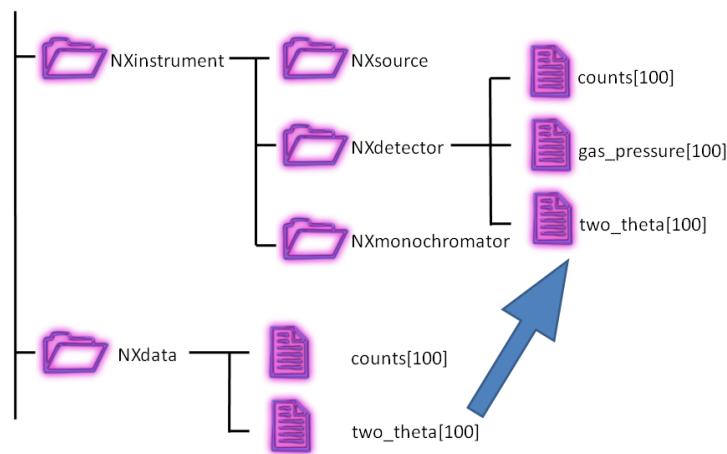


Fig. 1: Linking in a NeXus file

NeXus links are HDF5 hard links with an additional `target` attribute. The `target` attribute is added¹ for NeXus to distinguish the HDF5 path to the *original*² dataset. The value of the `target` attribute is the HDF5 path³ 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:

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 address³ 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 link⁴ 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?

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

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

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

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

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

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

```
1 EXTERNAL_LINK "data" {
2   TARGETFILE "/dls/i22/data/2012/sm7594-1/i22-69201-Pilatus2M.h5"
3   TARGETPATH "entry/instrument/detector/data"
4 }
```

Note

The NAPI code⁵ 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 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

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>

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*).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.

Inheritance in NeXus

Single inheritance is supported in NeXus for both base classes and application definitions (which are described below). Extending a base class or application definition inherits the properties and objects of the parent class or definition, such as groups, fields, attributes, and symbol tables. These properties and objects of the parent can then be overridden by the subclass. Only single inheritance is allowed (you can only inherit from a single parent).

Use the `@extends` attribute of a definition to indicate which definition is being subclassed. Base classes should extend from **NXObject** unless they are being subclassed.

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*.

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.)

⁷ see: <https://github.com/nexusformat/definitions/issues/397>

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
<i>NXnote</i>	to store additional information
<i>NXlog</i>	information that is time-stamped
<i>NXparameters</i>	parameters for processing or analysis
<i>NXcollection</i>	to store <i>any</i> 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.

Like base classes, NeXus supports *Inheritance in NeXus* 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.

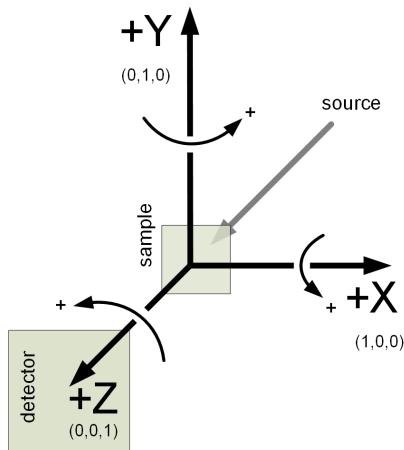


Fig. 2: NeXus coordinate system, as viewed from detector

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

```
1 sample:NXsample  
2 transforms:NXtransformations
```

(continues on next page)

(continued from previous page)

```

3     rotation_angle
4         @transformation_type=rotation
5             @vector=0,1,0
6             @offset=0,0,0
7             @depends_on=.
8     chi
9         @transformation_type=rotation
10            @vector=0,0,1
11            @offset=0,0,0
12            @depends_on=rotation_angle
13     phi
14         @transformation_type=rotation
15            @vector=0,1,0
16            @offset=0,0,0
17            @depends_on=chi
18     depends_on
19         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>).

```

1 OFF
2 # cube.off
3 # A cube

4
5 8 6 12
6 1.0 0.0 1.0
7 0.0 1.0 1.0
8 -1.0 0.0 1.0
9 0.0 -1.0 1.0
10 1.0 0.0 0.0
11 0.0 1.0 0.0
12 -1.0 0.0 0.0
13 0.0 -1.0 0.0
14 4 0 1 2 3
15 4 7 4 0 3
16 4 4 5 1 0
17 4 5 6 2 1
18 4 3 2 6 7
19 4 6 5 4 7

```

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

```

1 shape : NXoff_geometry
2 @NX_class = "NXoff_geometry"
3 vertices =
4   1.0, 0.0, 1.0
5   0.0, 1.0, 1.0
6   -1.0, 0.0, 1.0
7   0.0, -1.0, 1.0
8   1.0, 0.0, 0.0
9   0.0, 1.0, 0.0
10  -1.0, 0.0, 0.0

```

(continues on next page)

(continued from previous page)

```

11   0.0, -1.0, 0.0
12 faces =
13   0, 4, 8, 12, 16, 20
14 winding_order =
15   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.

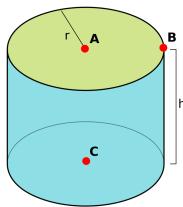


Fig. 3: Cylinder definition with three vertices

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://mestas.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:

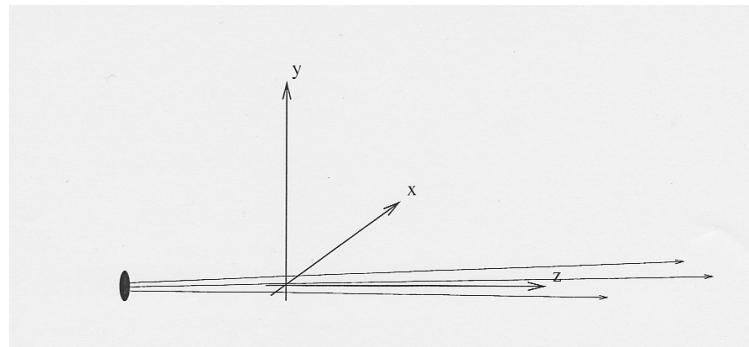


Fig. 4: The McStas Coordinate System

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](#).

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.

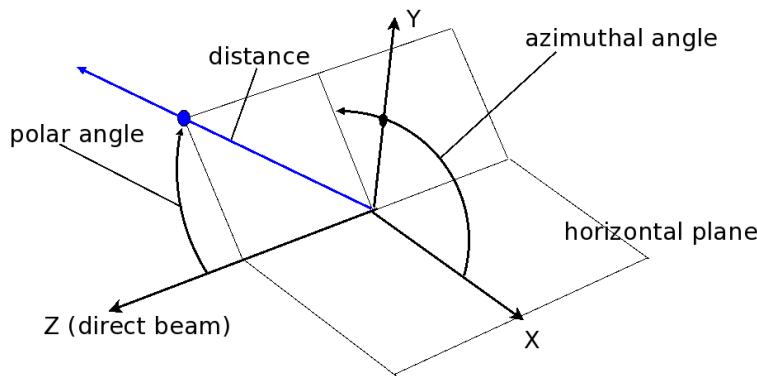


Fig. 5: NeXus Simple (Spherical Polar) Coordinate System

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 512×512 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

```

1 entry:NXentry
2   @default = data
3   instrument:NXinstrument
4     source:NXsource
5     ....
6     detector:NXdetector
7       data:NX_INT32[512,512]
8     sample:NXsample
9     control:NXmonitor
10    data:NXdata
11      @signal = data
12      @axes = [".", "."]
13      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

```
1   entry:NXentry
2     @default = data
3     reduction:NXprocess
4       program_name = "pyDataProc2010"
5       version = "1.0a"
6       input:NXparameters
7         filename = "sn2013287.nxs"
8       sample:NXsample
9       data:NXdata
10      @signal = data
11      @axes = "."
12      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 fluorescence data) was linked.

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

NeXus Multi Method Hierarchy

```

1   entry:NXentry
2     @default = data
3     user:NXuser
4     sample:NXsample
5     instrument:NXinstrument
6       SASdet:NXdetector
7         data:[,]
8         fluordet:NXdetector
9           data:[,]
10        large_area:NXdetector
11          data:[,]
12        SAS:NXsubentry
13          definition = "NXsas"
14          instrument:NXinstrument
15            detector:NXdetector
16              data --> /entry/instrument/SASdet/data
17          data:NXdata
18            data --> /entry/instrument/SASdet/data
19        Fluo:NXsubentry
20          definition = "NXfluo"
21          instrument:NXinstrument
22            detector --> /entry/instrument/fluordet/data
23            detector2 --> /entry/instrument/large_area/data
24          data:NXdata
25            @signal = detector
26            @axes = [".", ".."]
27            detector --> /entry/instrument/fluordet/data
28          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

```

1 entry:NXentry
2   @default = data
3   instrument:NXinstrument
4     detector:NXdetector
5       data[NP]
6   sample:NXsample
7     rotation_angle[NP]
8   control:NXmonitor
9     data[NP]
10  data:NXdata
11    @signal = "data"
12    @axes = "rotation_angle"
13    @rotation_angle_indices = 0
14    data --> /entry/instrument/detector/data
15    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

```

1 entry:NXentry
2   instrument:NXinstrument
3     detector:NXdetector
4       data:[NP,xsize,ysize]
5   sample:NXsample
6     rotation_angle[NP]
7   control:NXmonitor
8     data[NP]
9   data:NXdata
10    @signal = "data"
11    @axes = ["rotation_angle", ".", "."]
12    @rotation_angle_indices = 0

```

(continues on next page)

(continued from previous page)

```

13   data --> /entry/instrument/detector/data
14   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 multiple 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

```

1   entry:NXentry
2     @default = data
3     instrument:NXinstrument
4       detector:NXdetector
5         data[NP]
6         polar_angle[NP]
7         name
8       sample:NXsample
9         name
10        rotation_angle[NP]
11        chi[NP]
12        phi[NP]
13        h[NP]
14        k[NP]
15        l[NP]
16       control:NXmonitor
17         data[NP]
18       data:NXdata
19         @signal = data
20         @axes = "h"
21         @h_indices = 0
22         @k_indices = 0
23         @l_indices = 0
24         @chi_indices = 0
25         @phi_indices = 0
26         @polar_angle_indices = 0
27         @rotation_angle_indices = 0
28       data --> /entry/instrument/detector/data
29       rotation_angle --> /entry/sample/rotation_angle
30       chi --> /entry/sample/chi
31       phi --> /entry/sample/phi
32       polar_angle --> /entry/instrument/detector/polar_angle
33       h --> /entry/sample/h
34       k --> /entry/sample/k
35       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 $\log(I_0/I)$, we show them separately here in two different `NXdata` groups to demonstrate that such things are possible. Note that the length of the 1-D energy array is `NE` while the length of the 1-D temperature array is `NT`

NeXus Multi-parameter scan: XAS

```

1   entry:NXentry
2     @default = "I_data"
3     instrument:NXinstrument
4       I:NXdetector
5         data:NX_NUMBER[NE,NT]
6           energy --> /entry/monochromator/energy
7             temperature --> /entry/sample/temperature
8       I0:NXdetector
9         data:NX_NUMBER[NE,NT]
10        energy --> /entry/monochromator/energy
11          temperature --> /entry/sample/temperature
12     sample:NXsample
13       temperature:NX_NUMBER[NT]
14     monochromator:NXmonochromator
15       energy:NX_NUMBER[NE]
16     I_data:NXdata
17       @signal = "data"
18       @axes = ["energy", "temperature"]
19       @energy_indices = 0
20       @temperature_indices = 0
21       data --> /entry/instrument/I/data
22       energy --> /entry/monochromator/energy
23         temperature --> /entry/sample/temperature
24     I0_data:NXdata
25       @signal = data
26       @axes = ["energy", "temperature"]
27       @energy_indices = 0
28       @temperature_indices = 0
29       data --> /entry/instrument/I00/data
30       energy --> /entry/monochromator/energy
31         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 [NX-event_data](#) structure of the same name.

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

```
1 entry:NXentry
2     positioners:NXcollection
3         mxx:NXpositioner
4         mzz:NXpositioner
5         sgu:NXpositioner
6         ttv:NXpositioner
7         hugo:NXpositioner
8         ....
9     scalars:NXcollection
10        title (dataset)
11        lieselotte (dataset)
12        ...
13     detectors:NXcollection
14        Pilatus:NXdata
15        MX-45:NXdata
16        ....
```

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 names¹ 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*.

NXDL group and field names

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

- Recommended* names³
 - 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)² 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*):

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

The length should be limited to no more than 63 characters (historically imposed by the HDF4 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.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.

```
1 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.

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.

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.

² <https://en.wikipedia.org/wiki/ASCII>

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](#).

The data stored in NeXus fields must be *readback* values. This means values as read from the detector, other hardware, etc. There are occasions where it is sensible to store the target value the variable was supposed to have. In such cases, the *target* value is stored with a name built by appending `_set` to the NeXus (readback) field name.

Consider this example:

```
1 temperature
2 temperature_set
```

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

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

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

Reserved prefixes

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

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_identifier	attributes fields	reserved for use by silx reserved for unique identifier in groups	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	<i>NXtransformations</i>	end points of the motions that start with DATASET
_errors	<i>NXdata</i>	uncertainties (a.k.a., errors)
_increment	<i>NXtransformations</i>	intended average range through which the corresponding axis moves during the exposure of a frame
_indices	<i>NXdata</i>	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 <i>NXdetector</i> <code>pixel_mask</code> .
_set	<i>target values</i>	Target value of DATASET
_weights		divide DATASET by these weights ⁴
_scaling_f	<i>NXdata</i>	Multiply DATASET by this factor ⁴
_offset	<i>NXdata</i>	Add this factor to DATASET ⁴

⁴ If DATASET_weights exists and has the same shape as the field, you are supposed to divide DATASET by the weights. Similarly, if DATASET_scaling_factor and/or DATASET_offset exist, apply this equation: (DATASET + DATASET_offset) * DATASET_scaling_factor

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 preceding 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:

```
1 beam_center_x
2     @variant=beam_center_x_refined
3 beam_center_x_refined
4     @variant=beam_center_x_initial_guess
5 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:

```
1 data
2 data_errors
3 beam_center_x
4 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:

```

1   * raw data = 0 1 2 3 4 5 6 7 8 9
2     size[1] = { 10 } // assume uniform overall array dimensions
3
4   * default stride:
5     stride[1] = { 1 }
6     offset[1] = { 0 }
7     for i:
8       result[i]:
9         0 1 2 3 4 5 6 7 8 9
10
11  * reverse stride:
12    stride[1] = { -1 }
13    offset[1] = { 9 }
14    for i:
15      result[i]:
16        9 8 7 6 5 4 3 2 1 0

```

Offset and Stride for 2D Data

```

1   * raw data = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
2     size[2] = { 4, 5 } // assume uniform overall array dimensions
3
4   * row major (C) stride:
5     stride[2] = { 5, 1 }
6     offset[2] = { 0, 0 }
7     for i:
8       for j:
9         result[i][j]:
10           0 1 2 3 4
11             5 6 7 8 9
12               10 11 12 13 14
13                 15 16 17 18 19
14
15   * column major (Fortran) stride:
16     stride[2] = { 1, 4 }
17     offset[2] = { 0, 0 }
18     for i:
19       for j:
20         result[i][j]:
21           0 4 8 12 16
22             1 5 9 13 17
23               2 6 10 14 18
24                 3 7 11 15 19
25
26   * "crazy reverse" row major (C) stride:
27     stride[2] = { -5, -1 }
28     offset[2] = { 4, 5 }
29     for i:

```

(continues on next page)

(continued from previous page)

```

30     for j:
31         result[i][j]:
32             19 18 17 16 15
33             14 13 12 11 10
34             9 8 7 6 5
35             4 3 2 1 0

```

Offset and Stride for 3D Data

```

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

```

(continues on next page)

(continued from previous page)

```

43      10 22 34 46 58
44
45      2 14 26 38 50
46      5 17 29 41 53
47      8 20 32 44 56
48      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 8601⁵ 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.

⁵ ISO 8601: <https://www.w3.org/TR/NOTE-datetime>

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”.

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.

⁶ The UDunits specification also includes instructions for derived units. At present, the contents of `NeXus units` attributes are not validated in data files.

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

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.

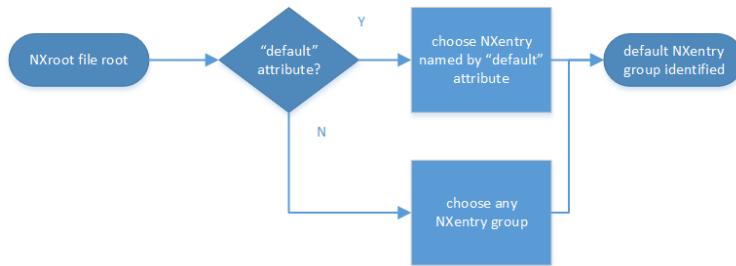


Fig. 6: Find plottable data: select the NXentry group

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.

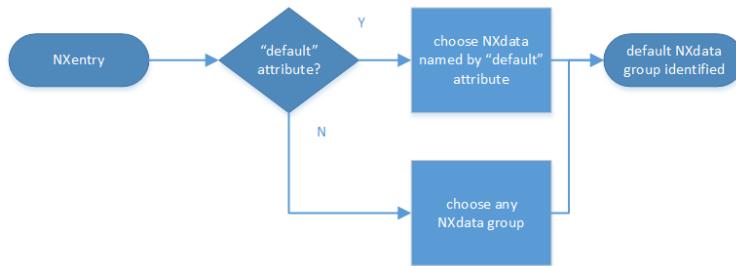


Fig. 7: Find plottable data: select the NXdata group

- 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.

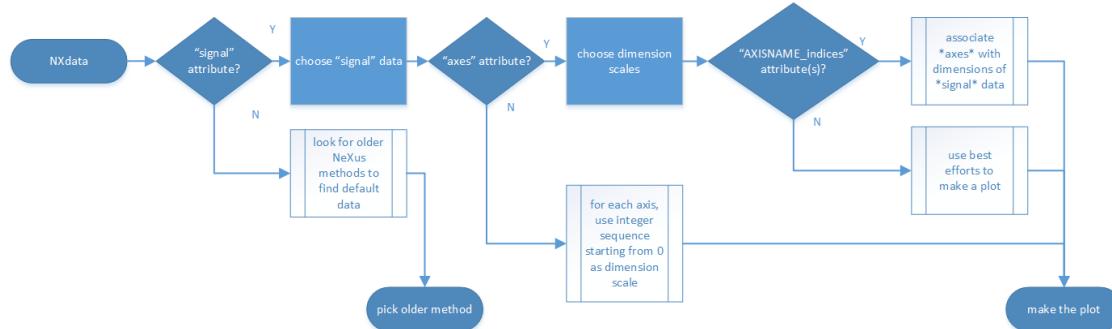


Fig. 8: Find plottable data: select the *signal* data

- 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.

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

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

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

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

- 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"

```

(continues on next page)

(continued from previous page)

```
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.

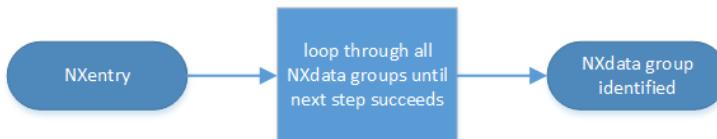


Fig. 10: Find plottable data: pick a `NXdata` group

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.

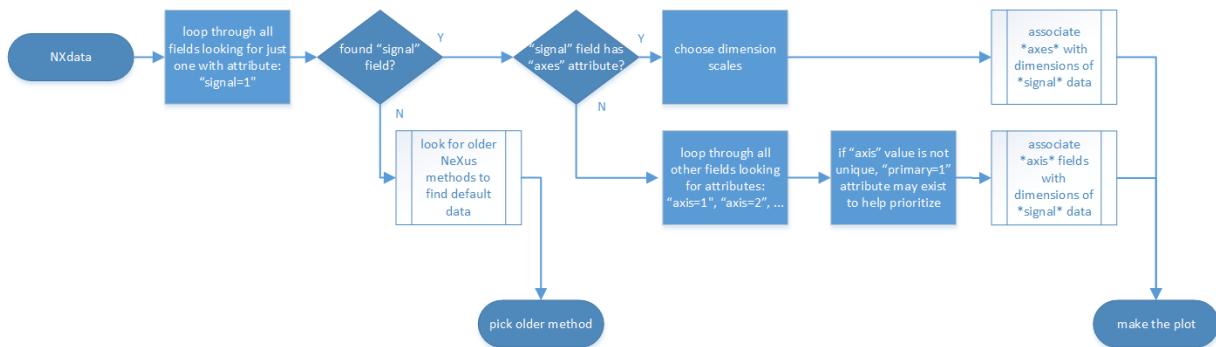


Fig. 11: Find plottable data: select the *signal* data

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.

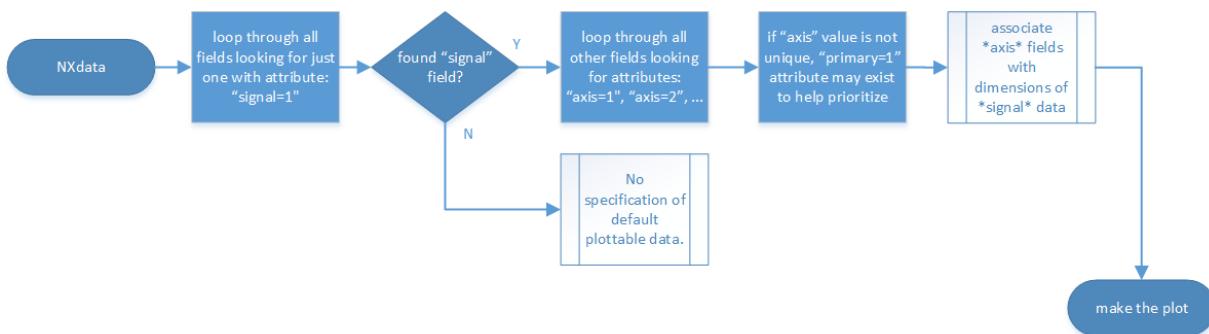


Fig. 14: Find plottable data: select the *signal* data

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 array⁸ 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 array⁸ 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 ([Page 52, 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.

```

1 datafile.hdf5:NeXus data file
2   @default="entry"
3     entry:NXentry
4       @default="data"
5         data:NXdata
6           @signal="counts"
7             @axes="mr"
8               @mr_indices=0
9               counts: float[100] --> the default dependent data
10              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).

```

1 datafile.hdf5:NeXus data file
2   @default="entry"
3     entry:NXentry
4       @default="data_2d"
5         data_2d:NXdata
6           @signal="data"
7             @axes=["time", "pressure"]
8               @pressure_indices=1
9               @temperature_indices=1
10              @time_indices=0
11              data: float[1000, 20]
12              pressure: float[20]
13              temperature: float[20]
14              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

```

1  data:NXdata
2    time_of_flight = 1500.0 1502.0 1504.0 ...
3    polar_angle = 15.0 15.6 16.2 ...
4    some_other_angle = 0.0 0.0 2.0 ...
5    data = 5 7 14 ...
6    @axes = ["polar_angle", "time_of_flight"]
7    @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

```

1  data:NXdata
2    time_of_flight = 1500.0 1502.0 1504.0 ...
3    @axis = 1
4    @primary = 1
5    polar_angle = 15.0 15.6 16.2 ...
6    @axis = 2
7    @primary = 1
8    some_other_angle = 0.0 0.0 2.0 ...
9    @axis = 1
10   data = 5 7 14 ...
11   @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 than 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

```

1 GROUP "entry" {
2     ATTRIBUTE "NX_class" {
3         DATATYPE H5T_STRING {
4             STRSIZE 7;
5             STRPAD H5T_STR_NULLPAD;
6             CSET H5T_CSET_ASCII;
7             CTYPE H5T_C_S1;
8         }
9         DATASPACE SCALAR
10        DATA {
11            () : "NXentry"
12        }
13    }
14    # ... group contents
15 }
```

h5dump of a NeXus field (HDF5 dataset)

```

1 DATASET "two_theta" {
2     DATATYPE H5T_IEEE_F64LE
3     DATASPACE SIMPLE { ( 31 ) / ( 31 ) }
4     DATA {
5         () : 17.9261, 17.9259, 17.9258, 17.9256, 17.9254, 17.9252,
6         (6) : 17.9251, 17.9249, 17.9247, 17.9246, 17.9244, 17.9243,
7         (12) : 17.9241, 17.9239, 17.9237, 17.9236, 17.9234, 17.9232,
```

(continues on next page)

(continued from previous page)

```

8      (18): 17.9231, 17.9229, 17.9228, 17.9226, 17.9224, 17.9222,
9      (24): 17.9221, 17.9219, 17.9217, 17.9216, 17.9214, 17.9213,
10     (30): 17.9211
11   }
12   ATTRIBUTE "units" {
13     DATATYPE H5T_STRING {
14       STRSIZE 7;
15       STRPAD H5T_STR_NULLPAD;
16       CSET H5T_CSET_ASCII;
17       CTYPE H5T_C_S1;
18     }
19     DATASPACE SCALAR
20     DATA {
21       (0): "degrees"
22     }
23   }
24   # ... other attributes
25 }
```

h5dump of a NeXus attribute

```

1 ATTRIBUTE "axes" {
2   DATATYPE H5T_STRING {
3     STRSIZE 9;
4     STRPAD H5T_STR_NULLPAD;
5     CSET H5T_CSET_ASCII;
6     CTYPE H5T_C_S1;
7   }
8   DATASPACE SCALAR
9   DATA {
10     (0): "two_theta"
11   }
12 }
```

h5dump of a NeXus link

```

1 # NeXus links have two parts in HDF5 files.
2
3 # The dataset is created in some group.
4 # A "target" attribute is added to indicate the HDF5 path to this dataset.
5
6 ATTRIBUTE "target" {
7   DATATYPE H5T_STRING {
8     STRSIZE 21;
9     STRPAD H5T_STR_NULLPAD;
10    CSET H5T_CSET_ASCII;
11    CTYPE H5T_C_S1;
12  }
13  DATASPACE SCALAR
14  DATA {
15    (0): "/entry/data/two_theta"
```

(continues on next page)

(continued from previous page)

```

16    }
17 }
18
19 # then, the hard link is created that refers to the original dataset
20 # (Since the name is "two_theta" in this example, it is understood that
21 # this link is created in a different HDF5 group than "/entry/data".)
22
23 DATASET "two_theta" {
24     HARDLINK "/entry/data/two_theta"
25 }
```

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.

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

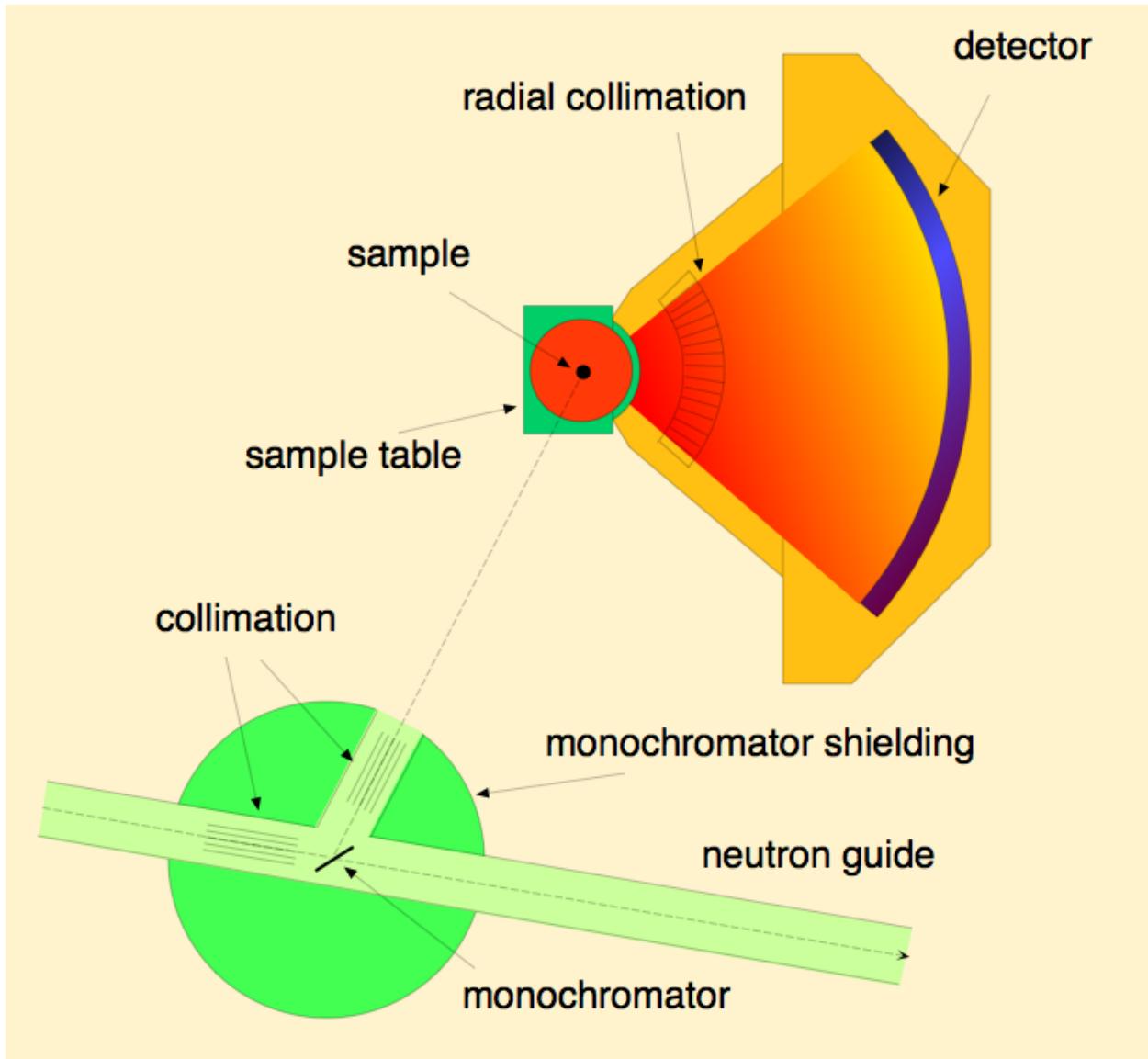


Fig. 15: The (fictional) WONI example powder diffractometer

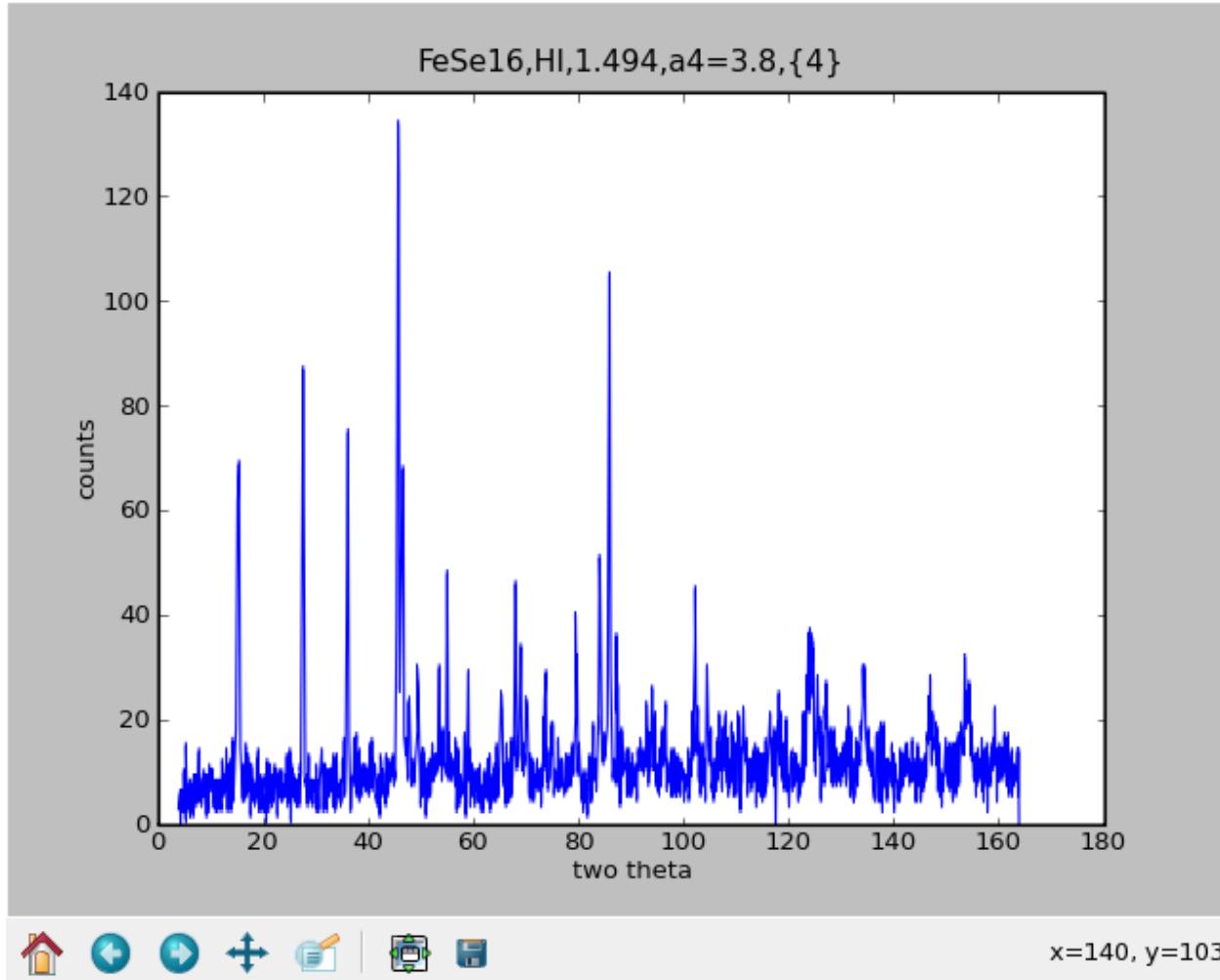


Fig. 16: Example Powder Diffraction Plot from (fictional) WONI at HYNES

Basic structure of a NeXus file

```

1 entry:NXentry
2   NXdata
3   NXinstrument
4   NXmonitor
5   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

```

1 entry:NXentry
2   NXdata
3   NXinstrument
4     monochromator:Nxcrystal
5       wavelength
6       d_spacing
7       rotation_angle
8       reflection
9       type
10      NXmonitor
11      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 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 than the first one. We need to map the data items which were collected in Step 1 into the NeXus hierarchy. A NeXus file hierarchy starts with an `NXentry` group. At this stage it is advisable to pull up the base class definition for `NXentry` and study it. The first thing you might notice is that `NXentry` contains a field named `title`. Reading the documentation, you quickly realize that this is a good place to store our title. So the first mapping has been found.

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

Note

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

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

```
1 sample_name = /NXentry/NXsample/name
2 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:

```

1 probe = /NXentry/NXinstrument/NXsource/probe
2 wavelength = /NXentry/NXinstrument/NXcrystal/wavelength
3 two theta of detector elements = /NXentry/NXinstrument/NXdetector/polar angle
4 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:

```

1 two theta of detector elements = /NXentry/NXdata/polar_angle
2 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
<i>title of measurement</i>	/NXentry/title
<i>sample name</i>	/NXentry/NXsample/name
<i>sample temperature</i>	/NXentry/NXsample/temperature
<i>monitor</i>	/NXentry/NXmonitor/data
<i>type of radiation probe</i>	/NXentry/NXinstrument/NXsource/probe
<i>wavelength</i> of radiation incident on sample	/NXentry/NXinstrument/NXcrystal/wavelength
<i>two theta</i> of detector elements	/NXentry/NXinstrument/NXdetector/polar_angle
<i>counts</i> for each detector element	/NXentry/NXinstrument/NXdetector/data
<i>two theta</i> of detector elements	/NXentry/NXdata/polar_angle
<i>counts</i> 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 `nxd1.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

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!--
3 # NeXus - Neutron and X-ray Common Data Format
4 #
5 # Copyright (C) 2008-2024 NeXus International Advisory Committee (NIAC)
6 #
7 # This library is free software; you can redistribute it and/or
8 # modify it under the terms of the GNU Lesser General Public
9 # License as published by the Free Software Foundation; either
10 # version 3 of the License, or (at your option) any later version.
11 #
12 # This library is distributed in the hope that it will be useful,
13 # but WITHOUT ANY WARRANTY; without even the implied warranty of
14 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
15 # Lesser General Public License for more details.
16 #
17 # You should have received a copy of the GNU Lesser General Public
18 # License along with this library; if not, write to the Free Software
19 # Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
20 #
21 # For further information, see https://www.nexusformat.org/
22 -->
23 <definition name="NX_template_" extends="NXobject" type="group"
24   category="application"
25   xmlns="http://definition.nexusformat.org/nxdl/3.1"
26   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
27   xsi:schemaLocation="http://definition.nexusformat.org/nxdl/3.1 ..//nxdl.xsd"
28 >
29   <doc>template for a NXDL application definition</doc>
30 </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:

```

1 <group type="NXdata">
2
3 </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:

```

1 <field name="polar_angle" type="NX_FLOAT units="NX_ANGLE">
2   <doc>Link to polar angle in /NXentry/NXinstrument/NXdetector</doc>
3   <dimensions rank="1">
4     <dim index="1" value="ndet"/>
5   </dimensions>
6 </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

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <?xml-stylesheet type="text/xsl" href="nxdlformat.xsl" ?>
3  <!--
4  # NeXus - Neutron and X-ray Common Data Format
5  #
6  # Copyright (C) 2008-2024 NeXus International Advisory Committee (NIAC)
7  #
8  # This library is free software; you can redistribute it and/or
9  # modify it under the terms of the GNU Lesser General Public
10 # License as published by the Free Software Foundation; either
11 # version 3 of the License, or (at your option) any later version.
12 #
13 # This library is distributed in the hope that it will be useful,
14 # but WITHOUT ANY WARRANTY; without even the implied warranty of
15 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
16 # Lesser General Public License for more details.
17 #
18 # You should have received a copy of the GNU Lesser General Public
19 # License along with this library; if not, write to the Free Software
20 # Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
21 #
22 # For further information, see http://www.nexusformat.org
23 -->
24 <definition name="NXmonopd" extends="NXobject" type="group"
25   category="application"
```

(continues on next page)

(continued from previous page)

```

26      xmlns="http://definition.nexusformat.org/nxdl/3.1"
27      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
28      xsi:schemaLocation="http://definition.nexusformat.org/nxdl/3.1 ..../nxdl.xsd"
29    >
30    <symbols>
31      <doc>
32        The symbol(s) listed here will be used below to coordinate datasets with the_
33        same shape.
34      </doc>
35      <symbol name="i">
36        <doc>i is the number of wavelengths</doc>
37      </symbol>
38      <symbol name="nDet">
39        <doc>Number of detectors</doc>
40      </symbol>
41    </symbols>
42    <doc>
43      Monochromatic Neutron and X-Ray Powder diffractometer
44
45      Instrument
46      definition for a powder diffractometer at a monochromatic neutron
47      or X-ray beam. This is both suited for a powder diffractometer
48      with a single detector or a powder diffractometer with a position
49      sensitive detector.
50    </doc>
51    <group type="NXentry">
52      <field name="title"/>
53      <field name="start_time" type="NX_DATE_TIME"/>
54      <field name="definition">
55        <doc> Official NeXus NXDL schema to which this file conforms </doc>
56        <enumeration>
57          <item value="NXmonopod"/>
58        </enumeration>
59      </field>
60      <group type="NXinstrument">
61        <group type="NXsource">
62          <field name="type"/>
63          <field name="name"/>
64          <field name="probe">
65            <enumeration>
66              <item value="neutron"/>
67              <item value="x-ray"/>
68              <item value="electron"/>
69            </enumeration>
70          </field>
71        </group>
72        <group type="NXcrystal">
73          <field name="wavelength" type="NX_FLOAT" units="NX_WAVELENGTH">
74            <doc>Optimum diffracted wavelength</doc>
75            <dimensions rank="1">
76              <dim index="1" value="i"/>
77            </dimensions>

```

(continues on next page)

(continued from previous page)

```

77      </field>
78  </group>
79  <group type="NXdetector">
80      <field name="polar_angle" type="NX_FLOAT" axis="1">
81          <dimensions rank="1">
82              <dim index="1" value="nDet" />
83          </dimensions>
84      </field>
85      <field name="data" type="NX_INT" signal="1">
86          <doc>
87              detector signal (usually counts) are already
88              corrected for detector efficiency
89          </doc>
90          <dimensions rank="1">
91              <dim index="1" value="nDet" />
92          </dimensions>
93      </field>
94  </group>
95 </group>
96  <group type="NXsample">
97      <field name="name">
98          <doc>Descriptive name of sample</doc>
99      </field>
100     <field name="rotation_angle" type="NX_FLOAT" units="NX_ANGLE">
101         <doc>
102             Optional rotation angle for the case when the powder diagram
103             has been obtained through an omega-2theta scan like from a
104             traditional single detector powder diffractometer
105         </doc>
106     </field>
107 </group>
108 <group type="NXmonitor">
109     <field name="mode">
110         <doc>
111             Count to a preset value based on either clock time (timer)
112             or received monitor counts (monitor).
113         </doc>
114         <enumeration>
115             <item value="monitor"/>
116             <item value="timer"/>
117         </enumeration>
118     </field>
119     <field name="preset" type="NX_FLOAT">
120         <doc>preset value for time or monitor</doc>
121     </field>
122     <field name="integral" type="NX_FLOAT" units="NX_ANY">
123         <doc>Total integral monitor counts</doc>
124     </field>
125 </group>
126 <group type="NXdata">
127     <link name="polar_angle" target="/NXentry/NXinstrument/NXdetector/polar_angle"
           <br/>>
```

(continues on next page)

(continued from previous page)

```

128     <doc>Link to polar angle in /NXentry/NXinstrument/NXdetector</doc>
129   </link>
130   <link name="data" target="/NXentry/NXinstrument/NXdetector/data">
131     <doc>Link to data in /NXentry/NXinstrument/NXdetector</doc>
132   </link>
133 </group>
134 </group>
135 </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 than 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'll have these positioners as arrays and assume that a default plot of *photodiode* vs. *ar*:

positioner arrays	detector arrays
ar, ay, dy	I0, I100, time, Epoch, photodiode

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

```

1 file.nxs: NeXus HDF5 data file
2   @default = "entry"
3   entry: NXentry
4     @NX_class = "NXentry"
5     @default = "data"
6     data: NXdata
7       @NX_class = "NXdata"
8       @signal = "photodiode"
9       @axes = "ar"
10      ar: NX_FLOAT[]
11      ay: NX_FLOAT[]
12      dy: NX_FLOAT[]
13      I0: NX_FLOAT[]
14      I100: NX_FLOAT[]
15      time: NX_FLOAT[]
16      Epoch: NX_FLOAT[]
17      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

```

1 entry: NXentry
2   @NX_class = NXentry
3   instrument: NXinstrument
4     @NX_class = NXinstrument
5     crystal: NXcrystal
6       @NX_class = NXcrystal
7       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

```

1 entry: NXentry
2   @NX_class = NXentry
3   instrument: NXinstrument
4     @NX_class = NXinstrument
5     monochromator: NXmonochromator
6       @NX_class = NXmonochromator
7       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

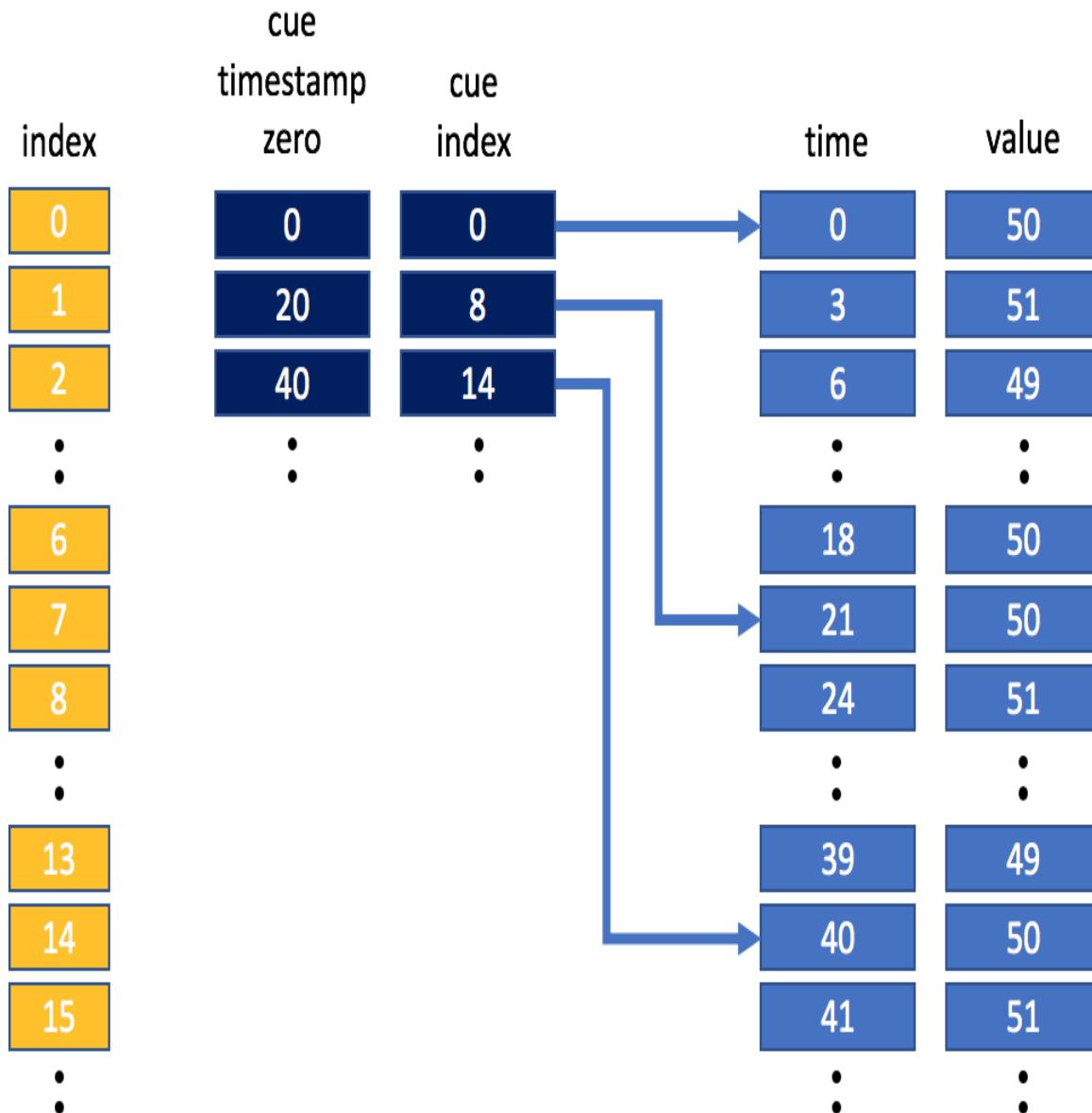
How should I store time-stamped data?

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?

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?

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

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/exampledatal>)

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?

Yes. When writing an application definition, you can add additional metadata to base classes that are not in the base class definition. This essentially sub-classes the original base class.

This is different than using the `extends` attribute, which can be used to make new base classes that inherit the properties of the parent classes, or new application definitions that use another as a starting point. For example the `NXelectron_detector` base class extends the `NXdetector` base class:

```
<definition category="base" name="NXelectron_detector" extends="NXdetector">
```

This is similar to how the application definition `NXdirectttof` extends the `NXtofraw` application definition:

```
<definition category="application" name="NXdirectttof" extends="NXtofraw">
```

Most NXDL files extend the `NXObject` base class (the base object of NeXus).

EXAMPLES OF WRITING AND READING NEXUS DATA FILES

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

2.1 Code Examples in Various Languages

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

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

2.1.1 HDF5 in C with libhdf5

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

Writing a simple NeXus file using native HDF5 commands in C

Note

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

```
1  /**
2   * This is an example how to write a valid NeXus file
3   * using the HDF-5 API alone. Ths structure which is
4   * going to be created is:
5   *
6   * scan:NXentry
7   *   data:NXdata
8   *     @signal = "counts"
9   *     @axes = "two_theta"
10  *     @two_theta_indices = 0
11  *     counts[]
12  *       @units="counts"
13  *     two_theta[]
14  *       @units="degrees"
15  *
```

(continues on next page)

(continued from previous page)

```

16 * WARNING: each of the HDF function below needs to be
17 * wrapped into something like:
18 *
19 * if((hdfid = H5function(...)) < 0){
20 *     handle error gracefully
21 * }
22 * I left the error checking out in order to keep the
23 * code clearer
24 *
25 * This also installs a link from /scan/data/two_theta to /scan/hugo
26 *
27 * Mark Koennecke, October 2011
28 */
29 #include <hdf5.h>
30 #include <stdlib.h>
31 #include <string.h>
32
33 static void write_string_attr(hid_t hid, const char* name, const char* value)
34 {
35     /* HDF-5 handles */
36     hid_t atts, atttype, attid;
37
38     atts = H5Screate(H5S_SCALAR);
39     atttype = H5Tcopy(H5T_C_S1);
40     H5Tset_size(atttype, strlen(value));
41     attid = H5Acreate(hid, name, atttype, atts, H5P_DEFAULT, H5P_DEFAULT);
42     H5Awrite(attid, atttype, value);
43     H5Sclose(atts);
44     H5Tclose(atttype);
45     H5Aclose(attid);
46 }
47
48 static void write_int_attr(hid_t hid, const char* name, int value)
49 {
50     /* HDF-5 handles */
51     hid_t atts, atttype, attid;
52
53     atts = H5Screate(H5S_SCALAR);
54     atttype = H5Tcopy(H5T_NATIVE_INT);
55     H5Tset_size(atttype, 1);
56     attid = H5Acreate(hid, name, atttype, atts, H5P_DEFAULT, H5P_DEFAULT);
57     H5Awrite(attid, atttype, &value);
58     H5Sclose(atts);
59     H5Tclose(atttype);
60     H5Aclose(attid);
61 }
62
63 #define LENGTH 400
64 int main(int argc, char *argv[])
65 {
66     float two_theta[LENGTH];
67     int counts[LENGTH], i, rank;
```

(continues on next page)

(continued from previous page)

```

68  /* HDF-5 handles */
69  hid_t fid, fapl, gid;
70  hid_t datatype, dataspace, dataprop, dataid;
71  hsize_t dim[1], maxdim[1];
72
73
74
75  /* create some data: nothing NeXus or HDF-5 specific */
76  for(i = 0; i < LENGTH; i++){
77      two_theta[i] = 10. + .1*i;
78      counts[i] = (int)(1000 * ((float)random()/(float)RAND_MAX));
79  }
80  dim[0] = LENGTH;
81  maxdim[0] = LENGTH;
82  rank = 1;
83
84
85
86  /*
87   * open the file. The file attribute forces normal file
88   * closing behaviour down HDF-5's throat
89   */
90  fapl = H5Pcreate(H5P_FILE_ACCESS);
91  H5Pset_fclose_degree(fapl,H5F_CLOSE_STRONG);
92  fid = H5Fcreate("NXfile.h5", H5F_ACC_TRUNC, H5P_DEFAULT,fapl);
93  H5Pclose(fapl);
94
95
96  /*
97   * create scan:NXentry
98   */
99  gid = H5Gcreate(fid, "scan",H5P_DEFAULT,H5P_DEFAULT,H5P_DEFAULT);
100
101
102  /* store the NX_class attribute. Notice that you
103   * have to take care to close those hids after use
104   */
105  write_string_attr(gid, "NX_class", "NXentry");
106
107
108  /*
109   * same thing for data:Nxdata in scan:NXentry.
110   */
111  gid = H5Gcreate(fid, "/scan/data",H5P_DEFAULT,H5P_DEFAULT,H5P_DEFAULT);
112  write_string_attr(gid, "NX_class", "Nxdata");
113
114
115  /*
116   * define axes.
117   */
118  write_string_attr(gid, "signal", "counts");
119  write_string_attr(gid, "axes", "two_theta");
120  write_int_attr(gid, "two_theta_indices", 0);
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149

```

(continues on next page)

(continued from previous page)

```

120 * store the counts dataset
121 */
122 dataspace = H5Screate_simple(rank,dim,maxdim);
123 datatype = H5Tcopy(H5T_NATIVE_INT);
124 dataprop = H5Pcreate(H5P_DATASET_CREATE);
125 dataid = H5Dcreate(gid,"counts",datatype,dataspace,H5P_DEFAULT,dataprop,H5P_DEFAULT);
126 H5Dwrite(dataid, datatype, H5S_ALL, H5S_ALL, H5P_DEFAULT, counts);
127 H5Sclose(dataspace);
128 H5Tclose(datatype);
129 H5Pclose(dataprop);
130 /*
131 * set the units attribute
132 */
133 write_string_attr(dataid, "units", "counts");
134
135 H5Dclose(dataid);

136 /*
137 * store the two_theta dataset
138 */
139 dataspace = H5Screate_simple(rank,dim,maxdim);
140 datatype = H5Tcopy(H5T_NATIVE_FLOAT);
141 dataprop = H5Pcreate(H5P_DATASET_CREATE);
142 dataid = H5Dcreate(gid,"two_theta",datatype,dataspace,H5P_DEFAULT,dataprop,H5P_
143 DEFAULT);
144 H5Dwrite(dataid, datatype, H5S_ALL, H5S_ALL, H5P_DEFAULT, two_theta);
145 H5Sclose(dataspace);
146 H5Tclose(datatype);
147 H5Pclose(dataprop);

148 /*
149 * set the units attribute
150 */
151 write_string_attr(dataid, "units", "degrees");

152 /*
153 * set the target attribute for linking
154 */
155 write_string_attr(dataid, "target", "/scan/data/two_theta");

156 H5Dclose(dataid);

157 /*
158 * make a link in /scan to /scan/data/two_theta, thereby
159 * renaming two_theta to hugo
160 */
161 H5Glink(fid,H5G_LINK_HARD,"/scan/data/two_theta","/scan/hugo");

162 /*
163 * close the file
164 */
165 H5Fclose(fid);

```

(continues on next page)

(continued from previous page)

171

}

Reading a simple NeXus file using native HDF5 commands in C

```

1  /**
2   * Reading example for reading NeXus files with plain
3   * HDF-5 API calls. This reads out counts and two_theta
4   * out of the file generated by nxh5write.
5   *
6   * WARNING: I left out all error checking in this example.
7   * In production code you have to take care of those errors
8   *
9   * Mark Koennecke, October 2011
10  */
11 #include <hdf5.h>
12 #include <stdlib.h>
13
14 int main(int argc, char *argv[])
15 {
16     float *two_theta = NULL;
17     int *counts = NULL, rank, i;
18     hid_t fid, dataid, fapl;
19     hsize_t *dim = NULL;
20     hid_t dataspace, memdataspace;
21
22     /*
23      * Open file, thereby enforcing proper file close
24      * semantics
25     */
26     fapl = H5Pcreate(H5P_FILE_ACCESS);
27     H5Pset_fclose_degree(fapl,H5F_CLOSE_STRONG);
28     fid = H5Fopen("NXfile.h5", H5F_ACC_RDONLY,fapl);
29     H5Pclose(fapl);
30
31     /*
32      * open and read the counts dataset
33     */
34     dataid = H5Dopen(fid,"/scan/data/counts",H5P_DEFAULT);
35     dataspace = H5Dget_space(dataid);
36     rank = H5Sget_simple_extent_ndims(dataspace);
37     dim = malloc(rank*sizeof(hsize_t));
38     H5Sget_simple_extent_dims(dataspace, dim, NULL);
39     counts = malloc(dim[0]*sizeof(int));
40     memdataspace = H5Tcopy(H5T_NATIVE_INT32);
41     H5Dread(dataid,memdataspace,H5S_ALL, H5S_ALL,H5P_DEFAULT, counts);
42     H5Dclose(dataid);
43     H5Sclose(dataspace);
44     H5Tclose(memdataspace);
45
46     /*
47      * open and read the two_theta data set

```

(continues on next page)

(continued from previous page)

```

48 */
49 dataid = H5Dopen(fid, "/scan/data/two_theta", H5P_DEFAULT);
50 dataspace = H5Dget_space(dataid);
51 rank = H5Sget_simple_extent_ndims(dataspace);
52 dim = malloc(rank*sizeof(hsize_t));
53 H5Sget_simple_extent_dims(dataspace, dim, NULL);
54 two_theta = malloc(dim[0]*sizeof(float));
55 memdataspace = H5Tcopy(H5T_NATIVE_FLOAT);
56 H5Dread(dataid,memdataspace,H5S_ALL, H5S_ALL,H5P_DEFAULT, two_theta);
57 H5Dclose(dataid);
58 H5Sclose(dataspace);
59 H5Tclose(memdataspace);

60
61
62
63 H5Fclose(fid);

64
65 for(i = 0; i < dim[0]; i++){
66     printf("%8.2f %10d\n", two_theta[i], counts[i]);
67 }
68
69 }

```

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.

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

¹ `nexusformat`: <https://nexpy.github.io/nexpy/>

² `h5py`: <https://www.h5py.org/>

³ `SPEC`: <http://certif.com/spec.html>

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 NXdata group called `mr_scan` 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`

```

1 #!/usr/bin/env python
2 """Writes a NeXus HDF5 file using h5py and numpy"""
3
4 from pathlib import Path
5 from re import X
6 import numpy
7
8 from nexusformat.nexus import NXdata, NXentry, NXfield, nxopen
9
10 print("Write a NeXus HDF5 file")
11 fileName = "simple_example_basic.nexus.hdf5"
12
13 # load data from two column format
14 data_filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
15 data = numpy.loadtxt(data_filename).T
16 mr_arr = data[0]
17 i00_arr = numpy.asarray(data[1], "int32")
18
19 # create the HDF5 NeXus file
20 with nxopen(fileName, "w") as f:
21
22     # create the NXentry group
23     f["entry"] = NXentry()
24     f["entry/title"] = "1-D scan of I00 v. mr"
25
26     # create the NXdata group
27     x = NXfield(mr_arr, name="mr", units="degrees", long_name="USAXS mr (degrees)")
```

(continues on next page)

(continued from previous page)

```

28     y = NXfield(i00_arr, name="I00", units="counts",
29                  long_name="USAXS I00 (counts)")
30     f["entry/mr_scan"] = NXdata(y, x)
31
32 print("wrote file:", fileName)

```

h5py

```

1 #!/usr/bin/env python
2 """Writes a NeXus HDF5 file using h5py and numpy"""
3
4 from pathlib import Path
5 import datetime
6 import h5py # HDF5 support
7 import numpy
8
9 print("Write a NeXus HDF5 file")
10 fileName = "simple_example_basic.nexus.hdf5"
11 timestamp = datetime.datetime.now().astimezone().isoformat()
12
13 # load data from two column format
14 data_filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
15 data = numpy.loadtxt(data_filename).T
16 mr_arr = data[0]
17 i00_arr = numpy.asarray(data[1], "int32")
18
19 # create the HDF5 NeXus file
20 with h5py.File(fileName, "w") as f:
21     # point to the default data to be plotted
22     f.attrs["default"] = "entry"
23     # give the HDF5 root some more attributes
24     f.attrs["file_name"] = fileName
25     f.attrs["file_time"] = timestamp
26     f.attrs["instrument"] = "APS USAXS at 32ID-B"
27     f.attrs["creator"] = "simple_example_basic_write.py"
28     f.attrs["NeXus_version"] = "4.3.0"
29     f.attrs["HDF5_Version"] = h5py.version.hdf5_version
30     f.attrs["h5py_version"] = h5py.version.version
31
32 # create the NXentry group
33 nxentry = f.create_group("entry")
34 nxentry.attrs["NX_class"] = "NXentry"
35 nxentry.attrs["default"] = "mr_scan"
36 nxentry.create_dataset("title", data="1-D scan of I00 v. mr")
37
38 # create the NXentry group
39 nxdata = nxentry.create_group("mr_scan")
40 nxdata.attrs["NX_class"] = "NXdata"
41 nxdata.attrs["signal"] = "I00" # Y axis of default plot
42 nxdata.attrs["axes"] = "mr" # X axis of default plot
43 nxdata.attrs["mr_indices"] = [
44     0,

```

(continues on next page)

(continued from previous page)

```

45 ] # use "mr" as the first dimension of I00
46
47 # X axis data
48 ds = nxdata.create_dataset("mr", data=mr_arr)
49 ds.attrs["units"] = "degrees"
50 ds.attrs["long_name"] = "USAXS mr (degrees)" # suggested X axis plot label
51
52 # Y axis data
53 ds = nxdata.create_dataset("I00", data=i00_arr)
54 ds.attrs["units"] = "counts"
55 ds.attrs["long_name"] = "USAXS I00 (counts)" # suggested Y axis plot label
56
57 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 as `i00`.

`simple_example_basic_read.py`: Read a NeXus HDF5 file using Python

nexusformat

```

1 #!/usr/bin/env python
2 """Reads NeXus HDF5 files using nexusformat and prints the contents"""
3
4 from nexusformat.nexus import nxopen
5
6 fileName = "simple_example_basic.nexus.hdf5"
7 with nxopen(fileName) as f:
8     print(f.tree)

```

h5py

```

1 #!/usr/bin/env python
2 """Reads NeXus HDF5 files using h5py and prints the contents"""
3
4 import h5py # HDF5 support
5
6 fileName = "simple_example_basic.nexus.hdf5"
7 with h5py.File(fileName, "r") as f:
8     for item in f.attrs.keys():
9         print(item + ":", f.attrs[item])
10    mr = f["/entry/mr_scan/mr"]

```

(continues on next page)

(continued from previous page)

```
11     i00 = f["/entry/mr_scan/I00"]
12     print("%s\t%s\t%" % ("#", "mr", "I00"))
13     for i in range(len(mr)):
14         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`

```
1 file_name: simple_example_basic.nexus.hdf5
2 file_time: 2010-10-18T17:17:04-0500
3 creator: simple_example_basic_write.py
4 HDF5_Version: 1.8.5
5 NeXus_version: 4.3.0
6 h5py_version: 1.2.1
7 instrument: APS USAXS at 32ID-B
8 #   mr  I00
9 0   17.9261 1037
10 1   17.9259 1318
11 2   17.9258 1704
12 3   17.9256 2857
13 4   17.9254 4516
14 5   17.9252 9998
15 6   17.9251 23819
16 7   17.9249 31662
17 8   17.9247 40458
18 9   17.9246 49087
19 10  17.9244 56514
20 11  17.9243 63499
21 12  17.9241 66802
22 13  17.9239 66863
23 14  17.9237 66599
24 15  17.9236 66206
25 16  17.9234 65747
26 17  17.9232 65250
27 18  17.9231 64129
28 19  17.9229 63044
29 20  17.9228 60796
30 21  17.9226 56795
31 22  17.9224 51550
32 23  17.9222 43710
33 24  17.9221 29315
34 25  17.9219 19782
35 26  17.9217 12992
36 27  17.9216 6622
37 28  17.9214 4198
38 29  17.9213 2248
39 30  17.9211 1321
```

downloads

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

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

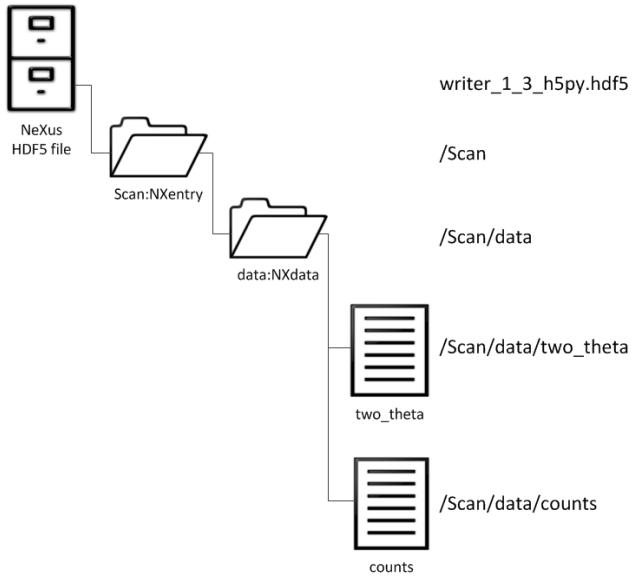


Fig. 1: Simple Example

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

```
1 #!/usr/bin/env python
2 """
3 Writes the simplest NeXus HDF5 file using h5py
4
5 Uses method accepted at 2014NIAC
6 according to the example from Figure 1.3
7 in the Introduction chapter
8 """
9
10 from pathlib import Path
11
12 import numpy
13
14 from nexusformat.nexus import NXdata, NXentry, NXfield, nxopen
15
16 filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
17 buffer = numpy.loadtxt(filename).T
18 tthData = buffer[0]
19 countsData = numpy.asarray(buffer[1], "int32")
20
21 with nxopen("simple_example_write1.hdf5", "w") as f: # create the NeXus file
22     f["Scan"] = NXentry()
23     tth = NXfield(tthData, name="two_theta", units="degrees")
24     counts = NXfield(countsData, name="counts", units="counts")
25     f["Scan/data"] = NXdata(counts, tth)
```

h5py

```
1 #!/usr/bin/env python
2 """
3 Writes the simplest NeXus HDF5 file using h5py
4
5 Uses method accepted at 2014NIAC
6 according to the example from Figure 1.3
7 in the Introduction chapter
8 """
9
10 from pathlib import Path
11 import h5py
12 import numpy
13
14 filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
15 buffer = numpy.loadtxt(filename).T
16 tthData = buffer[0] # float[]
17 countsData = numpy.asarray(buffer[1], "int32") # int[]
18
19 with h5py.File("simple_example_write1.hdf5", "w") as f: # create the HDF5 NeXus file
20     # since this is a simple example, no attributes are used at this point
```

(continues on next page)

(continued from previous page)

```

21
22     nxentry = f.create_group("Scan")
23     nxentry.attrs["NX_class"] = "NXentry"
24
25     nxdata = nxentry.create_group("data")
26     nxdata.attrs["NX_class"] = "NXdata"
27     nxdata.attrs["signal"] = "counts"
28     nxdata.attrs["axes"] = "two_theta"
29     nxdata.attrs["two_theta_indices"] = [
30         @,
31     ]
32
33     tth = nxdata.create_dataset("two_theta", data=tthData)
34     tth.attrs["units"] = "degrees"
35
36     counts = nxdata.create_dataset("counts", data=countsData)
37     counts.attrs["units"] = "counts"

```

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

Note

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

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

```

1 Scan:NXentry
2   @NX_class = "NXentry"
3   data:NXdata
4     @NX_class = "NXdata"
5     @axes = "two_theta"
6     @signal = "counts"
7     @two_theta_indices = [0]
8     counts:NX_INT32[31] = [1037, 1318, 1704, ..., 1321]
9       @units = "counts"
10    two_theta:NX_FLOAT64[31] = [17.92608, 17.92591, 17.92575, ..., 17.92108]
11      @units = "degrees"

```

As the data files in these examples become more complex, you will appreciate the information density provided by `punx tree`.

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

downloads

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

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

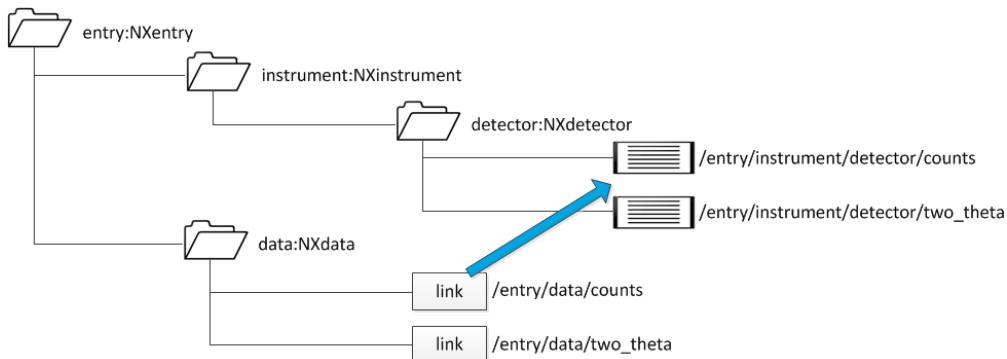


Fig. 2: h5py example showing linking in a NeXus file

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

nexusformat

```

1  #!/usr/bin/env python
2  """
3  Writes a simple NeXus HDF5 file using h5py with links
4  according to the example from Figure 2.1 in the Design chapter
5  """
6
7  from pathlib import Path
8
9  import numpy

```

(continues on next page)

(continued from previous page)

```

10
11 from nexusformat.nexus import (NXdata, NXdetector, NXentry, NXfield,
12                                 NXinstrument, NXlink, nxopen)
13
14 filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
15 buffer = numpy.loadtxt(filename).T
16 tthData = buffer[0] # float[]
17 countsData = numpy.asarray(buffer[1], "int32") # int[]
18
19 with nxopen("simple_example_write2.hdf5", "w") as f: # create the HDF5 NeXus
20     # file
21     f["entry"] = NXentry()
22     f["entry/instrument"] = NXinstrument()
23     f["entry/instrument/detector"] = NXdetector()
24
25     # store the data in the NXdetector group
26     f["entry/instrument/detector/two_theta"] = NXfield(tthData, units="degrees"
27             )
27     f["entry/instrument/detector/counts"] = NXfield(countsData, units="counts")
28
29     f["entry/data"] = NXdata(NXlink("/entry/instrument/detector/counts"),
30                             NXlink("/entry/instrument/detector/two_theta"))
31     f["entry/data"].set_default()

```

h5py

```

1#!/usr/bin/env python
2"""
3    Writes a simple NeXus HDF5 file using h5py with links
4    according to the example from Figure 2.1 in the Design chapter
5"""
6
7from pathlib import Path
8import h5py
9import numpy
10
11filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
12buffer = numpy.loadtxt(filename).T
13tthData = buffer[0] # float[]
14countsData = numpy.asarray(buffer[1], "int32") # int[]
15
16with h5py.File("simple_example_write2.hdf5", "w") as f: # create the HDF5
17    # NeXus file
18    f.attrs["default"] = "entry"
19
20    nxentry = f.create_group("entry")
21    nxentry.attrs["NX_class"] = "NXentry"
22    nxentry.attrs["default"] = "data"
23
24    nxinstrument = nxentry.create_group("instrument")
25    nxinstrument.attrs["NX_class"] = "NXinstrument"

```

(continues on next page)

(continued from previous page)

```

26     nxdetector = nxinstrument.create_group("detector")
27     nxdetector.attrs["NX_class"] = "NXdetector"
28
29     # store the data in the NXdetector group
30     ds_tth = nxdetector.create_dataset("two_theta", data=tthData)
31     ds_tth.attrs["units"] = "degrees"
32     ds_counts = nxdetector.create_dataset("counts", data=countsData)
33     ds_counts.attrs["units"] = "counts"
34
35     # create the NXdata group to define the default plot
36     nxdata = nxentry.create_group("data")
37     nxdata.attrs["NX_class"] = "NXdata"
38     nxdata.attrs["signal"] = "counts"
39     nxdata.attrs["axes"] = "two_theta"
40     nxdata.attrs["two_theta_indices"] = [
41         0,
42     ]
43
44     source_addr = "/entry/instrument/detector/two_theta" # existing data
45     target_addr = "two_theta" # new location
46     ds_tth.attrs["target"] = source_addr # a NeXus API convention for links
47     nxdata[target_addr] = f[source_addr] # hard link
48     # nxdata._id.link(source_addr, target_addr, h5py.h5g.LINK_HARD)
49
50     source_addr = "/entry/instrument/detector/counts" # existing data
51     target_addr = "counts" # new location
52     ds_counts.attrs["target"] = source_addr # a NeXus API convention for links
53     nxdata[target_addr] = f[source_addr] # hard link
54     # 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.

```

1  @default = "entry"
2  entry:NXentry
3  @NX_class = "NXentry"
4  @default = "data"
5  data:NXdata

```

(continues on next page)

(continued from previous page)

```

6      @NX_class = "NXdata"
7      @axes = "two_theta"
8      @signal = "counts"
9      @two_theta_indices = [0]
10     counts --> /entry/instrument/detector/counts
11     two_theta --> /entry/instrument/detector/two_theta
12 instrument:NXinstrument
13     @NX_class = "NXinstrument"
14     detector:NXdetector
15       @NX_class = "NXdetector"
16       counts:NX_INT32[31] = [1037, 1318, 1704, '...', 1321]
17         @target = "/entry/instrument/detector/counts"
18         @units = "counts"
19       two_theta:NX_FLOAT64[31] = [17.92608, 17.92591, 17.92575, '...', 17.92108]
20         @target = "/entry/instrument/detector/two_theta"
21         @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 <i>simple_example_write2</i>
nexusformat/simple_example_write2.py	nexusformat code to write example <i>simple_example_write2</i>
simple_example_write2.hdf5	NeXus file written by this code
simple_example_write2_h5dump.txt	<i>h5dump</i> analysis of the NeXus file
simple_example_write2_structure.txt	<i>punx tree</i> 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:

```

1 angles:float64[31] = [17.926079999999999, '...', 17.92108]
2   @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:

```
1 entry:NXentry
2   instrument:NXinstrument
3   detector:NXdetector
4   counts:NX_INT32[31] = [1037, '...', 1321]
5     @units = counts
6   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:

```
1 entry:NXentry
2 @default = data
3   instrument --> file="external_counts.hdf5", path="/entry/instrument"
4   data:NXdata
5     @signal = counts
6     @axes = "two_theta"
7     @two_theta = 0
8     counts --> file="external_counts.hdf5", path="/entry/instrument/detector/counts"
9     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.

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>

external_example_write.py: Write using HDF5 external links

nexusformat

```

1 #!/usr/bin/env python
2 """
3 Writes a NeXus HDF5 file using h5py with links to data in other HDF5 files.
4
5 This example is based on ``writer_2_1``.
6 """
7
8 from pathlib import Path
9
10 import h5py
11 import numpy
12
13 from nexusformat.nexus import (NXdata, NXdetector, NXentry, NXfield,
14                                NXinstrument, NXlink, nxopen)
15
16 FILE_HDF5_MASTER = "external_master.hdf5"
17 FILE_HDF5_ANGLES = "external_angles.hdf5"
18 FILE_HDF5_COUNTS = "external_counts.hdf5"
19
20 # -----
21
22 # get some data
23 filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
24 buffer = numpy.loadtxt(filename).T
25 tthData = buffer[0] # float[]
26 countsData = numpy.asarray(buffer[1], "int32") # int[]
27
28 # put the angle data in an external (non-NeXus) HDF5 data file
29 with h5py.File(FILE_HDF5_ANGLES, "w") as f:
30     ds = f.create_dataset("angles", data=tthData)
31     ds.attrs["units"] = "degrees"
32
33 # put the detector counts in an external HDF5 data file
34 # with *incomplete* NeXus structure (no NXdata group)
35 with nxopen(FILE_HDF5_COUNTS, "w") as f:
36     f["entry"] = NXentry()
37     f["entry/instrument"] = NXinstrument()
38     f["entry/instrument/detector"] = NXdetector()
39     f["entry/instrument/detector/counts"] = NXfield(countsData, units="counts")
40     f["entry/instrument/detector/two_theta"] = NXlink("/angles",
41                                                 FILE_HDF5_ANGLES)
42
43 # create a master NeXus HDF5 file
44 with nxopen(FILE_HDF5_MASTER, "w") as f:
45     f["entry"] = NXentry()
46     counts = NXlink("/entry/instrument/detector/counts", FILE_HDF5_COUNTS,
47                     name="counts")
48     two_theta = NXlink("/angles", FILE_HDF5_ANGLES, name="two_theta")
49     f["entry/data"] = NXdata(counts, two_theta)
50     f["entry/data"].set_default()

```

(continues on next page)

(continued from previous page)

```
51     f["entry/instrument"] = NXlink("/entry/instrument", FILE_HDF5_COUNTS)
```

h5py

```
1 #!/usr/bin/env python
2 """
3 Writes a NeXus HDF5 file using h5py with links to data in other HDF5 files.
4
5 This example is based on ``writer_2_1``.
6 """
7
8 from pathlib import Path
9 import h5py
10 import numpy
11
12 FILE_HDF5_MASTER = "external_master.hdf5"
13 FILE_HDF5_ANGLES = "external_angles.hdf5"
14 FILE_HDF5_COUNTS = "external_counts.hdf5"
15
16 # -----
17
18 # get some data
19 filename = str(Path(__file__).absolute().parent.parent / "simple_example.dat")
20 buffer = numpy.loadtxt(filename).T
21 tthData = buffer[0] # float[]
22 countsData = numpy.asarray(buffer[1], "int32") # int[]
23
24 # put the angle data in an external (non-NeXus) HDF5 data file
25 with h5py.File(FILE_HDF5_ANGLES, "w") as f:
26     ds = f.create_dataset("angles", data=tthData)
27     ds.attrs["units"] = "degrees"
28
29 # put the detector counts in an external HDF5 data file
30 # with *incomplete* NeXus structure (no NXdata group)
31 with h5py.File(FILE_HDF5_COUNTS, "w") as f:
32     nxentry = f.create_group("entry")
33     nxentry.attrs["NX_class"] = "NXentry"
34     nxinstrument = nxentry.create_group("instrument")
35     nxinstrument.attrs["NX_class"] = "NXinstrument"
36     nxdetector = nxinstrument.create_group("detector")
37     nxdetector.attrs["NX_class"] = "NXdetector"
38     ds = nxdetector.create_dataset("counts", data=countsData)
39     ds.attrs["units"] = "counts"
40     # link the "two_theta" data stored in separate file
41     local_addr = nxdetector.name + "/two_theta"
42     f[local_addr] = h5py.ExternalLink(FILE_HDF5_ANGLES, "/angles")
43
44 # create a master NeXus HDF5 file
45 with h5py.File(FILE_HDF5_MASTER, "w") as f:
46     f.attrs["default"] = "entry"
47     nxentry = f.create_group("entry")
48     nxentry.attrs["NX_class"] = "NXentry"
```

(continues on next page)

(continued from previous page)

```

49     nxentry.attrs["default"] = "data"
50     nxdata = nxentry.create_group("data")
51     nxdata.attrs["NX_class"] = "NXdata"
52
53     # link in the signal data
54     local_addr = "/entry/data/counts"
55     external_addr = "/entry/instrument/detector/counts"
56     f[local_addr] = h5py.ExternalLink(FILE_HDF5_COUNTS, external_addr)
57     nxdata.attrs["signal"] = "counts"
58
59     # link in the axes data
60     local_addr = "/entry/data/two_theta"
61     f[local_addr] = h5py.ExternalLink(FILE_HDF5_ANGLES, "/angles")
62     nxdata.attrs["axes"] = "two_theta"
63     nxdata.attrs["two_theta_indices"] = [
64         0,
65     ]
66
67     local_addr = "/entry/instrument"
68     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	<i>h5dump</i> analysis of <i>external_angles.hdf5</i>
external_angles.hdf5	HDF5 file written by <i>external_example_write</i>
external_angles_structure.txt	<i>punx tree</i> analysis of <i>external_angles.hdf5</i>
external_counts_h5dump.txt	<i>h5dump</i> analysis of <i>external_counts.hdf5</i>
external_counts.hdf5	HDF5 file written by <i>external_example_write</i>
external_counts_structure.txt	<i>punx tree</i> analysis of <i>external_counts.hdf5</i>
external_example_write.py	<i>h5py</i> code to write external linking examples
nexusformat/external_example_write.py	<i>nexusformat</i> code to write external linking examples
external_master_h5dump.txt	<i>h5dump</i> analysis of <i>external_master.hdf5</i>
external_master.hdf5	NeXus file written by <i>external_example_write</i>
external_master_structure.txt	<i>punx tree</i> analysis of <i>external_master.hdf5</i>

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 have 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

```

1  from pathlib import Path
2  from nexusformat.nexus import nxopen

```

(continues on next page)

(continued from previous page)

```

3   filename = str(
4     Path(__file__).absolute().parent.parent
5     / "simple_example_basic"
6     / "simple_example_basic.nexus.hdf5"
7   )
8
9   with nxopen(filename) as f:
10    # find the default NXdata group
11    nx_data = f.get_default()
12    signal = nx_data.nxsignal
13    axes = nx_data.nxaxes[0]
14
15  nx_data.plot() # plot the data using Matplotlib
16
17 print(f"file: {f.nxfilename}")
18 print(f"signal: {signal.nxname}")
19 print(f"axes: {axes.nxname}")
20 print(f"{axes.nxname} {signal.nxname}")
21 for x, y in zip(axes, signal):
22   print(x, y)

```

h5py

```

1  from pathlib import Path
2  import h5py
3
4  filename = str(
5    Path(__file__).absolute().parent.parent
6    / "simple_example_basic"
7    / "simple_example_basic.nexus.hdf5"
8  )
9  with h5py.File(filename, "r") as nx:
10   # find the default NXentry group
11   nx_entry = nx[nx.attrs["default"]]
12   # find the default NXdata group
13   nx_data = nx_entry[nx_entry.attrs["default"]]
14   # find the signal field
15   signal = nx_data[nx_data.attrs["signal"]]
16   # find the axes field(s)
17   attr_axes = nx_data.attrs["axes"]
18   if isinstance(attr_axes, (set, tuple, list)):
19     # but check that attr_axes only describes 1-D data
20     if len(attr_axes) == 1:
21       attr_axes = attr_axes[0]
22     else:
23       raise ValueError(f"expected 1-D data but @axes={attr_axes}")
24   axes = nx_data[attr_axes]
25
26   print(f"file: {nx.filename}")
27   print(f"signal: {signal.name}")
28   print(f"axes: {axes.name}")
29   print(f"{axes.name} {signal.name}")

```

(continues on next page)

(continued from previous page)

```

30     for x, y in zip(axes, signal):
31         print(x, y)

```

Output from `reader_attributes_trail.py` is shown next.

Output from `reader_attributes_trail.py`

```

1 file: simple_example_basic.nexus.hdf5
2 signal: /entry/mr_scan/I00
3 axes: /entry/mr_scan/mr
4 /entry/mr_scan/mr /entry/mr_scan/I00
5 17.92608 1037
6 17.92591 1318
7 17.92575 1704
8 17.92558 2857
9 17.92541 4516
10 17.92525 9998
11 17.92508 23819
12 17.92491 31662
13 17.92475 40458
14 17.92458 49087
15 17.92441 56514
16 17.92425 63499
17 17.92408 66802
18 17.92391 66863
19 17.92375 66599
20 17.92358 66206
21 17.92341 65747
22 17.92325 65250
23 17.92308 64129
24 17.92291 63044
25 17.92275 60796
26 17.92258 56795
27 17.92241 51550
28 17.92225 43710
29 17.92208 29315
30 17.92191 19782
31 17.92175 12992
32 17.92158 6622
33 17.92141 4198
34 17.92125 2248
35 17.92108 1321

```

downloads

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

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

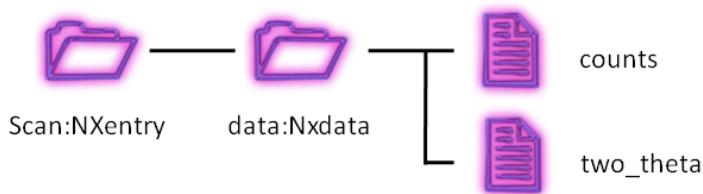


Fig. 3: data structure of the simple example

```

1 /entry:NXentry
2   /mr_scan:NXdata
3     /mr : float64[31]
4     /I00 : int32[31]
  
```

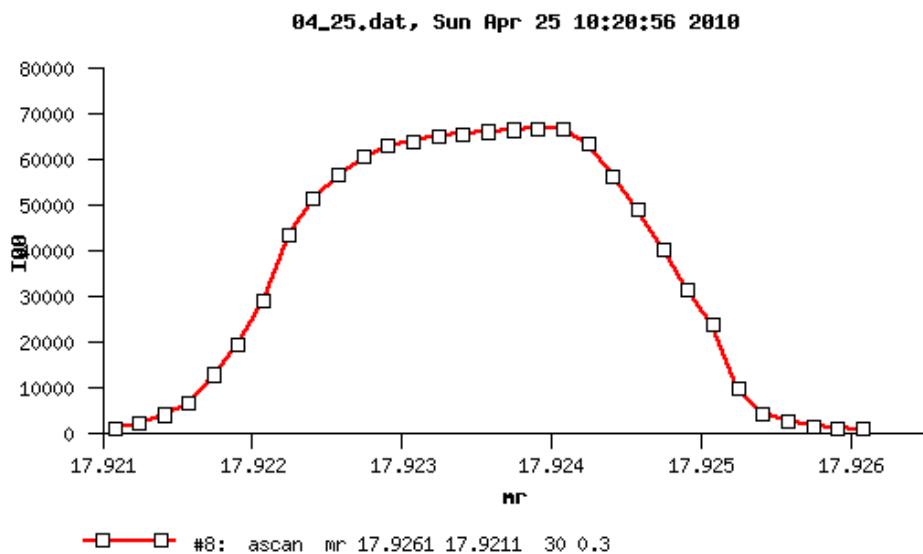


Fig. 4: plot of the simple example data

Simple example values

```

1 17.92608 1037
2 17.92591 1318
3 17.92575 1704
  
```

(continues on next page)

(continued from previous page)

4	17.92558	2857
5	17.92541	4516
6	17.92525	9998
7	17.92508	23819
8	17.92491	31662
9	17.92475	40458
10	17.92458	49087
11	17.92441	56514
12	17.92425	63499
13	17.92408	66802
14	17.92391	66863
15	17.92375	66599
16	17.92358	66206
17	17.92341	65747
18	17.92325	65250
19	17.92308	64129
20	17.92291	63044
21	17.92275	60796
22	17.92258	56795
23	17.92241	51550
24	17.92225	43710
25	17.92208	29315
26	17.92191	19782
27	17.92175	12992
28	17.92158	6622
29	17.92141	4198
30	17.92125	2248
31	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*.

1	17.92608	1037
2	17.92591	1318
3	17.92575	1704
4	17.92558	2857
5	17.92541	4516
6	17.92525	9998
7	17.92508	23819
8	17.92491	31662

(continues on next page)

(continued from previous page)

9	17.92475	40458
10	17.92458	49087
11	17.92441	56514
12	17.92425	63499
13	17.92408	66802
14	17.92391	66863
15	17.92375	66599
16	17.92358	66206
17	17.92341	65747
18	17.92325	65250
19	17.92308	64129
20	17.92291	63044
21	17.92275	60796
22	17.92258	56795
23	17.92241	51550
24	17.92225	43710
25	17.92208	29315
26	17.92191	19782
27	17.92175	12992
28	17.92158	6622
29	17.92141	4198
30	17.92125	2248
31	17.92108	1321

writing data

basic_writer.m: Write a NeXus HDF5 file using Matlab

```

1 % Writes a NeXus HDF5 file using matlab
2
3 disp 'Write a NeXus HDF5 file'
4 filename = 'prj_test.nexus.hdf5';
5 timestamp = '2010-10-18T17:17:04-0500';
6
7 % read input data
8 A = load('input.dat');
9 mr = A(:,1);
10 I00 = int32(A(:,2));
11
12 % clear out old file, if it exists
13
14 delete(filename);
15
16 % using the simple h5 interface, there is no way to create a group without
17 % first creating a dataset; creating the dataset creates all intervening
18 % groups.
19
20 % store x
21 h5create(filename,'/entry/mr_scan/mr',[length(mr)]);
22 h5write(filename,'/entry/mr_scan/mr',mr);
23 h5writeatt(filename,'/entry/mr_scan/mr','units','degrees');
24 h5writeatt(filename,'/entry/mr_scan/mr','long_name','USAXS mr (degrees)');

```

(continues on next page)

(continued from previous page)

```

25 % store y
26 h5create(filename,'/entry/mr_scan/I00',[length(I00)],'DataType','int32');
27 h5write(filename,'/entry/mr_scan/I00',I00);
28 h5writeatt(filename,'/entry/mr_scan/I00','units','counts');
29 h5writeatt(filename,'/entry/mr_scan/I00','long_name','USAXS I00 (counts)');
30
31 % indicate that we are plotting y vs. x
32 h5writeatt(filename,'/','default','entry');
33 h5writeatt(filename,'/entry','default','mr_scan');
34 h5writeatt(filename,'/entry/mr_scan','signal','I00');
35 h5writeatt(filename,'/entry/mr_scan','axes','mr_scan');
36 h5writeatt(filename,'/entry/mr_scan','mr_scan_indices', int32(0));
37
38 % add NeXus metadata
39 h5writeatt(filename,'/','file_name',filename);
40 h5writeatt(filename,'/','file_time',timestamp);
41 h5writeatt(filename,'/','instrument','APS USAXS at 32ID-B');
42 h5writeatt(filename,'/','creator','basic_writer.m');
43 h5writeatt(filename,'/','NeXus_version','4.3.0');
44 h5writeatt(filename,'/','HDF5_Version','1.6'); % no 1.8 features used in this example
45 h5writeatt(filename,'/entry','NX_class','NXentry');
46 h5writeatt(filename,'/entry/mr_scan','NX_class','NXdata');
47
48
49
50 h5disp(filename);

```

reading data

basic_reader.m: Read a NeXus HDF5 file using Matlab

```

1 % Reads NeXus HDF5 file and print the contents
2
3 filename = 'prj_test.nexus.hdf5';
4 root = h5info(filename,'/');
5 attrs = root.Attributes;
6 for i = 1:length(attrs)
7     fprintf('%s: %s\n', attrs(i).Name, attrs(i).Value);
8 end
9 mr = h5read(filename,'/entry/mr_scan/mr');
10 i00 = h5read(filename, '/entry/mr_scan/I00');
11 fprintf('#\t%s\t%s\n','mr','I00');
12 for i = 1:length(mr)
13     fprintf('%d\t%g\t%d\n', i, mr(i), i00(i));
14 end

```

writing data file with links

writer_2_1.m: Write a NeXus HDF5 file with links

```

1 % Writes a simple NeXus HDF5 file with links
2 % according to the example from Figure 2.1 in the Design chapter

```

(continues on next page)

(continued from previous page)

```

3   filename = 'writer_2_1.hdf5';
4
5
6 % read input data
7 A = load('input.dat');
8 two_theta = A(:,1);
9 counts = int32(A(:,2));
10
11 % clear out old file, if it exists
12 delete(filename);
13
14 % store x
15 h5create(filename,'/entry/instrument/detector/two_theta',[length(two_theta)]);
16 h5write(filename,'/entry/instrument/detector/two_theta',two_theta);
17 h5writeatt(filename,'/entry/instrument/detector/two_theta','units','degrees');
18
19 % store y
20 h5create(filename,'/entry/instrument/detector/counts',[length(counts)],'DataType','int32
21 ↵');
21 h5write(filename,'/entry/instrument/detector/counts',counts);
22 h5writeatt(filename,'/entry/instrument/detector/counts','units','counts');
23
24 % create group NXdata with links to detector
25 % note: requires the additional file h5link.m
26 h5link(filename,'/entry/instrument/detector/two_theta','/entry/data/two_theta');
27 h5link(filename,'/entry/instrument/detector/counts','/entry/data/counts');
28
29 % indicate that we are plotting y vs. x
30 h5writeatt(filename,'/','default','entry');
31 h5writeatt(filename,'/entry','default','data');
32 h5writeatt(filename,'/entry/data','signal','counts');
33 h5writeatt(filename,'/entry/data','axes','two_theta');
34 h5writeatt(filename,'/entry/data','two_theta_indices',int32(0));
35
36 % add NeXus metadata
37 h5writeatt(filename,'/','file_name',filename);
38 h5writeatt(filename,'/','file_time',timestamp);
39 h5writeatt(filename,'/','instrument','APS USAXS at 32ID-B');
40 h5writeatt(filename,'/','creator','writer_2_1.m');
41 h5writeatt(filename,'/','NeXus_version','4.3.0');
42 h5writeatt(filename,'/','HDF5_Version','1.6'); % no 1.8 features used in this example
43 h5writeatt(filename,'/entry','NX_class','NXentry');
44 h5writeatt(filename,'/entry/instrument','NX_class','NXinstrument');
45 h5writeatt(filename,'/entry/instrument/detector','NX_class','NXdetector');
46 h5writeatt(filename,'/entry/data','NX_class','NXdata');
47
48 % show structure of the file that was created
49 h5disp(filename);

```

h5link.m: support module for creating NeXus-style HDF5 hard links

```

1 function h5link(filename, from, to)
2 %H5LINK Create link to an HDF5 dataset.
3 % H5LINK(FILENAME,SOURCE,TARGET) creates an HDF5 link from the
4 % dataset at location SOURCE to a dataset at location TARGET. All
5 % intermediate groups in the path to target are created.
6 %
7 % Example: create a link from /hello/world to /goodbye/world
8 %     h5create('myfile.h5','/hello/world',[100 200]);
9 %     h5link('myfile.h5','/hello/world','/goodbye/world');
10 %     hgdisp('myfile.h5');
11 %
12 % See also: h5create, h5read, h5write, h5info, h5disp
13 %
14 % split from and to into group/dataset
15 idx = strfind(from,'/');
16 from_path = from(1:idx(end)-1);
17 from_data = from(idx(end)+1:end);
18 idx = strfind(to,'/');
19 to_path = to(1:idx(end)-1);
20 to_data = to(idx(end)+1:end);
21 %
22 % open the HDF file
23 fid = H5F.open(filename,'H5F_ACC_RDWR','H5P_DEFAULT');
24 %
25 % create target group if it doesn't already exist
26 create_intermediate = H5P.create('H5P_LINK_CREATE');
27 H5P.set_create_intermediate_group(create_intermediate, 1);
28 try
29     H5G.create(fid,to_path,create_intermediate,'H5P_DEFAULT','H5P_DEFAULT');
30 catch
31 end
32 H5P.close(create_intermediate);
33 %
34 % open groups and create link
35 from_id = H5G.open(fid, from_path);
36 to_id = H5G.open(fid, to_path);
37 H5L.create_hard(from_id, from_data, to_id, to_data, 'H5P_DEFAULT','H5P_DEFAULT');
38 %
39 % close all
40 H5G.close(from_id);
41 H5G.close(to_id);
42 H5F.close(fid);
43 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
m	
basic_reader.m	reads the NeXus HDF5 file written by basic_writer.m
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*.

```

1 #include "napi.h"
2
3 int main()
4 {
5     int counts[50][1000], n_t=1000, n_p=50, dims[2], i;
6     float t[1000], phi[50];
7     NXhandle file_id;
8
9     /* Read in data using local routines to populate phi and counts
10    */
11    /* for example you may create a getdata() function and call
12    */
13    /*      getdata (n_t, t, n_p, phi, counts);
14    */
15    /* Open output file and output global attributes */
16    NXopen ("NXfile.nxs", NXACC_CREATE5, &file_id);
17    NXputattr (file_id, "user_name", "Joe Bloggs", 10, NX_CHAR);
18    /* Open top-level NXentry group */
19    NXmakegroup (file_id, "Entry1", "NXentry");
20    NXopengroup (file_id, "Entry1", "NXentry");
21    /* Open NXdata group within NXentry group */
22    NXmakegroup (file_id, "Data1", "NXdata");

```

(continues on next page)

(continued from previous page)

```

23   NXopengroup (file_id, "Data1", "NXdata");
24 /* Output time channels */
25   NXmakedata (file_id, "time_of_flight", NX_FLOAT32, 1, &n_t);
26   NXopendata (file_id, "time_of_flight");
27   NXputdata (file_id, t);
28   NXputattr (file_id, "units", "microseconds", 12, NX_CHAR);
29   NXclosedata (file_id);
30 /* Output detector angles */
31   NXmakedata (file_id, "polar_angle", NX_FLOAT32, 1, &n_p);
32   NXopendata (file_id, "polar_angle");
33   NXputdata (file_id, phi);
34   NXputattr (file_id, "units", "degrees", 7, NX_CHAR);
35   NXclosedata (file_id);
36 /* Output data */
37   dims[0] = n_t;
38   dims[1] = n_p;
39   NXmakedata (file_id, "counts", NX_INT32, 2, dims);
40   NXopendata (file_id, "counts");
41   NXputdata (file_id, counts);
42   i = 1;
43   NXputattr (file_id, "signal", &i, 1, NX_INT32);
44   NXputattr (file_id, "axes", "polar_angle:time_of_flight", 26, NX_CHAR);
45   NXclosedata (file_id);
46 /* Close NXentry and NXdata groups and close file */
47   NXclosegroup (file_id);
48   NXclosegroup (file_id);
49   NXclose (&file_id);
50   return;
51 }
```

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.

```

1 program WRITEDATA
2
3 include 'NAPIF.INC'
4 integer*4 status, file_id(NXHANDLESIZE), counts(1000,50), n_p, n_t, dims(2)
5 real*4 t(1000), phi(50)
6
```

(continues on next page)

(continued from previous page)

```

7 !Read in data using local routines
8     call getdata (n_t, t, n_p, phi, counts)
9 !Open output file
10    status = NXopen ('NXFILE.NXS', NXACC_CREATE, file_id)
11    status = NXputcharattr
12    +      (file_id, 'user', 'Joe Bloggs', 10, NX_CHAR)
13 !Open top-level NXentry group
14    status = NXmakegroup (file_id, 'Entry1', 'NXentry')
15    status = NXopengroup (file_id, 'Entry1', 'NXentry')
16 !Open NXdata group within NXentry group
17    status = NXmakegroup (file_id, 'Data1', 'NXdata')
18    status = NXopengroup (file_id, 'Data1', 'NXdata')
19 !Output time channels
20     status = NXmakedata
21     +      (file_id, 'time_of_flight', NX_FLOAT32, 1, n_t)
22     status = NXopendata (file_id, 'time_of_flight')
23     status = NXputdata (file_id, t)
24     status = NXputcharattr
25     +      (file_id, 'units', 'microseconds', 12, NX_CHAR)
26     status = NXclosedata (file_id)
27 !Output detector angles
28     status = NXmakedata (file_id, 'polar_angle', NX_FLOAT32, 1, n_p)
29     status = NXopendata (file_id, 'polar_angle')
30     status = NXputdata (file_id, phi)
31     status = NXputcharattr (file_id, 'units', 'degrees', 7, NX_CHAR)
32     status = NXclosedata (file_id)
33 !Output data
34     dims(1) = n_t
35     dims(2) = n_p
36     status = NXmakedata (file_id, 'counts', NX_INT32, 2, dims)
37     status = NXopendata (file_id, 'counts')
38     status = NXputdata (file_id, counts)
39     status = NXputattr (file_id, 'signal', 1, 1, NX_INT32)
40     status = NXputattr
41     +      (file_id, 'axes', 'polar_angle:time_of_flight', 26, NX_CHAR)
42     status = NXclosedata (file_id)
43 !Close NXdata and NXentry groups and close file
44     status = NXclosegroup (file_id)
45     status = NXclosegroup (file_id)
46     status = NXclose (file_id)
47
48 stop
49 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](#).

```

1 program WRITEDATA
2
3   use NXUmodule
4
5   type(NXhandle) :: file_id
6   integer, pointer :: counts(:,:)
7   real, pointer :: t(:, ), phi(:, )
8
9   !Use local routines to allocate pointers and fill in data
10  call getlocaldata (t, phi, counts)
11  !Open output file
12  if (NXopen ("NXfile.nxs", NXACC_CREATE, file_id) /= NX_OK) stop
13  if (NXUwriteglobals (file_id, user="Joe Bloggs") /= NX_OK) stop
14  !Set compression parameters
15  if (NXUsetcompress (file_id, NX_COMP_LZW, 1000) /= NX_OK) stop
16  !Open top-level NXentry group
17  if (NXUwritegroup (file_id, "Entry1", "NXentry") /= NX_OK) stop
18  !Open NXdata group within NXentry group
19  if (NXUwritegroup (file_id, "Data1", "NXdata") /= NX_OK) stop
20  !Output time channels
21  if (NXUwritedata (file_id, "time_of_flight", t, "microseconds") /= NX_OK) stop
22  !Output detector angles
23  if (NXUwritedata (file_id, "polar_angle", phi, "degrees") /= NX_OK) stop
24  !Output data
25  if (NXUwritedata (file_id, "counts", counts, "counts") /= NX_OK) stop
26  if (NXputattr (file_id, "signal", 1) /= NX_OK) stop
27  if (NXputattr (file_id, "axes", "polar_angle:time_of_flight") /= NX_OK) stop
28  !Close NXdata group
29  if (NXclosegroup (file_id) /= NX_OK) stop
30  !Close NXentry group
31  if (NXclosegroup (file_id) /= NX_OK) stop
32  !Close Nexus file
33  if (NXclose (file_id) /= NX_OK) stop
34
35 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

```

1 #!/usr/bin/python
2
3 import sys
4 import nxs
5 import numpy
6
7 a = numpy.zeros((2,3,4), dtype=numpy.int)

```

(continues on next page)

(continued from previous page)

```

8 val = 0
9 for i in range(2):
10     for j in range(3):
11         for k in range(4):
12             a[i,j,k] = val
13             val = val + 1
14
15 nf = nxs.open("simple3D.h5", "w5")
16
17 nf.makegroup("entry", "NXentry")
18 nf.opengroup("entry", "NXentry")
19
20 nf.makegroup("data", "NXdata")
21 nf.opengroup("data", "NXdata")
22 nf.putattr("signal", "test")
23
24 nf.makedata("test", 'int32', [2,3,4])
25 nf.opendata("test")
26 nf.putdata(a)
27 nf.closedata()
28
29 nf.closegroup() # NXdata
30 nf.closegroup() # NXentry
31
32 nf.close()
33
34 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.

¹ *NeXpy*: <http://nepy.github.io/nepy/>

¹ *silx view* : <http://www.silx.org/doc/silx/latest/applications/view.html>

² *silx* : <http://www.silx.org/doc/silx/latest/>

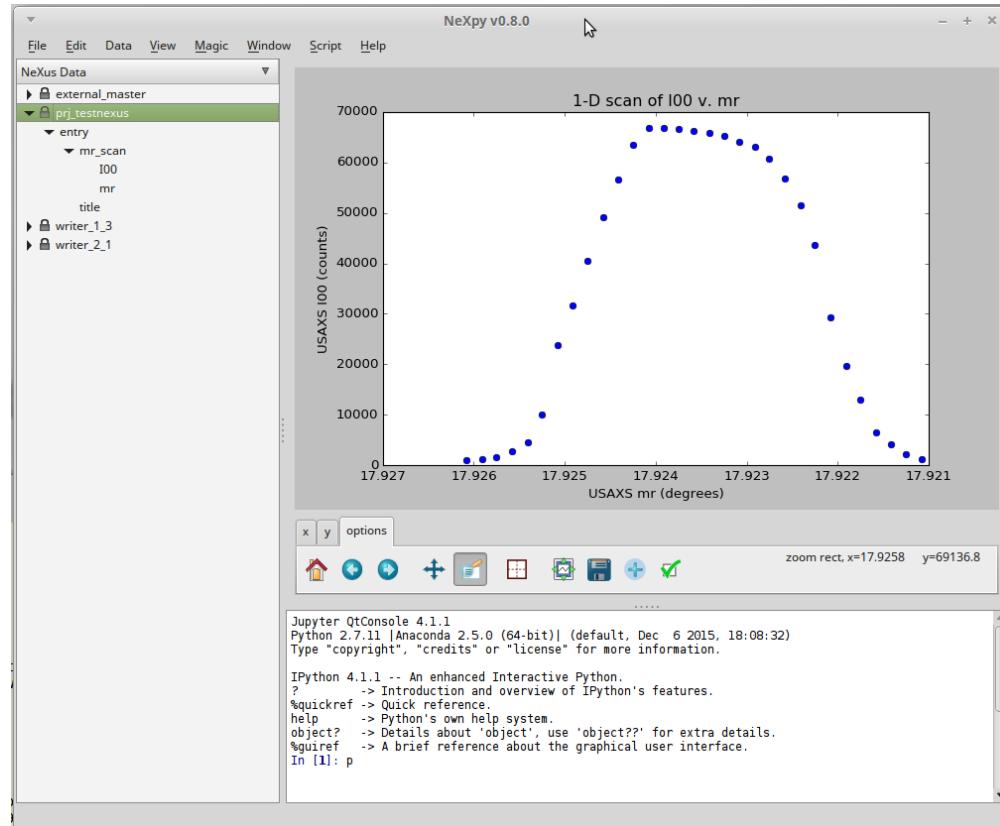


Fig. 5: plot the simple example using NeXpy

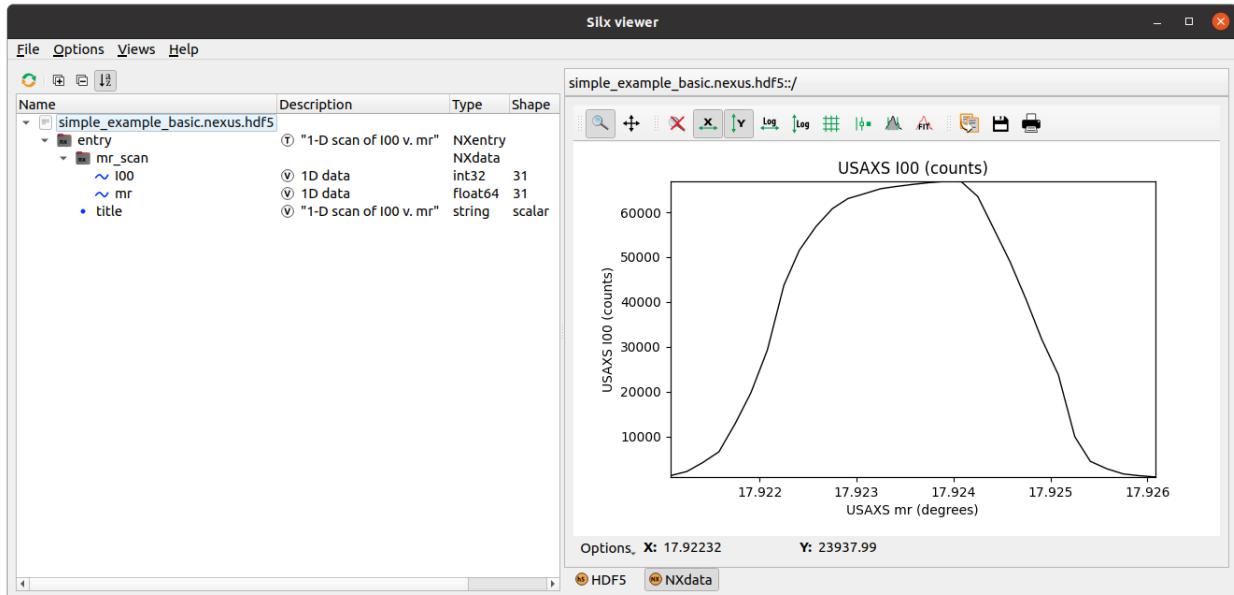


Fig. 6: plot the simple example using silx

2.2.3 View a NeXus HDF5 file with *punx tree*

The `punx tree` tool¹ 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](#)

```

1 simple3D.h5:NeXus data file
2 @NeXus_version = 4.1.0
3 @file_name = simple3D.h5
4 @HDF5_Version = 1.6.6
5 @file_time = 2011-11-18 17:26:27+0100
6 entry:NXentry
7   @NX_class = NXentry
8   data:NXdata
9     @NX_class = NXdata
10    test:NX_INT32[2,3,4] = __array
11      @signal = 1
12      __array = [
13        [
14          [0, 1, 2, 3]
15          [4, 5, 6, 7]
16          [8, 9, 10, 11]
17        ]
18        [
19          [12, 13, 14, 15]
20          [16, 17, 18, 19]
21          [20, 21, 22, 23]
22        ]
23      ]

```

2.2.4 View a NeXus HDF5 file with *h5dump*

The `h5dump` tool¹ provided as part of the HDF5 tool kit² 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](#)

```

1 HDF5 "simple3D.h5" {
2 GROUP "/" {
3   ATTRIBUTE "NeXus_version" {
4     DATATYPE H5T_STRING {
5       STRSIZE 5;
6       STRPAD H5T_STR_NULLTERM;
7       CSET H5T_CSET_ASCII;
8       CTYPE H5T_C_S1;
9     }
10    DATASPACE SCALAR
11    DATA {
12      (0): "4.1.0"
13    }
14  }
15  ATTRIBUTE "file_name" {
16    DATATYPE H5T_STRING {

```

(continues on next page)

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

² `punx` : <https://punx.readthedocs.io/>

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)

```

17     STRSIZE 11;
18     STRPAD H5T_STR_NULLTERM;
19     CSET H5T_CSET_ASCII;
20     CTYPE H5T_C_S1;
21   }
22   DATASPACE SCALAR
23   DATA {
24     (0): "simple3D.h5"
25   }
26 }
27 ATTRIBUTE "HDF5_Version" {
28   DATATYPE H5T_STRING {
29     STRSIZE 5;
30     STRPAD H5T_STR_NULLTERM;
31     CSET H5T_CSET_ASCII;
32     CTYPE H5T_C_S1;
33   }
34   DATASPACE SCALAR
35   DATA {
36     (0): "1.6.6"
37   }
38 }
39 ATTRIBUTE "file_time" {
40   DATATYPE H5T_STRING {
41     STRSIZE 24;
42     STRPAD H5T_STR_NULLTERM;
43     CSET H5T_CSET_ASCII;
44     CTYPE H5T_C_S1;
45   }
46   DATASPACE SCALAR
47   DATA {
48     (0): "2011-11-18 17:26:27+0100"
49   }
50 }
51 GROUP "entry" {
52   ATTRIBUTE "NX_class" {
53     DATATYPE H5T_STRING {
54       STRSIZE 7;
55       STRPAD H5T_STR_NULLTERM;
56       CSET H5T_CSET_ASCII;
57       CTYPE H5T_C_S1;
58     }
59     DATASPACE SCALAR
60     DATA {
61       (0): "NXentry"
62     }
63   }
64   GROUP "data" {
65     ATTRIBUTE "NX_class" {
66       DATATYPE H5T_STRING {
67         STRSIZE 6;
68         STRPAD H5T_STR_NULLTERM;
69       }
70     }
71   }
72 }
```

(continues on next page)

(continued from previous page)

```

69      CSET H5T_CSET_ASCII;
70      CTYPE H5T_C_S1;
71  }
72  DATASPACE SCALAR
73  DATA {
74      (0): "NXdata"
75  }
76 }
77 DATASET "test" {
78     DATATYPE H5T_STD_I32LE
79     DATASPACE SIMPLE { ( 2, 3, 4 ) / ( 2, 3, 4 ) }
80     DATA {
81         (0,0,0): 0, 1, 2, 3,
82         (0,1,0): 4, 5, 6, 7,
83         (0,2,0): 8, 9, 10, 11,
84         (1,0,0): 12, 13, 14, 15,
85         (1,1,0): 16, 17, 18, 19,
86         (1,2,0): 20, 21, 22, 23
87     }
88     ATTRIBUTE "signal" {
89         DATATYPE H5T_STD_I32LE
90         DATASPACE SCALAR
91         DATA {
92             (0): 1
93         }
94     }
95 }
96 }
97 }
98 }
99 }
```

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/exampledataltree/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

```

1 HDF5 "C:\Users\Pete\Documents\eclipse\NeXus\definitions\exampledata\IPNS\LRMECS\lrcs3701.
2   ↵nx5" {
3     GROUP "/Histogram1/data" {
4       DATASET "data" {
5         DATATYPE H5T_STD_I32LE
6         DATASPACE SIMPLE { ( 148, 750 ) / ( 148, 750 ) }
7       }
8       DATASET "polar_angle" {
9         DATATYPE H5T_IEEE_F32LE
10        DATASPACE SIMPLE { ( 148 ) / ( 148 ) }
11      }
12      DATASET "time_of_flight" {
13        DATATYPE H5T_IEEE_F32LE
14        DATASPACE SIMPLE { ( 751 ) / ( 751 ) }
15      }
16      DATASET "title" {
17        DATATYPE H5T_STRING {
18          STRSIZE 44;
19          STRPAD H5T_STR_NULLTERM;
20          CSET H5T_CSET_ASCII;
21          CTYPE H5T_C_S1;
22        }
23        DATASPACE SIMPLE { ( 1 ) / ( 1 ) }
24      }
25    }

```

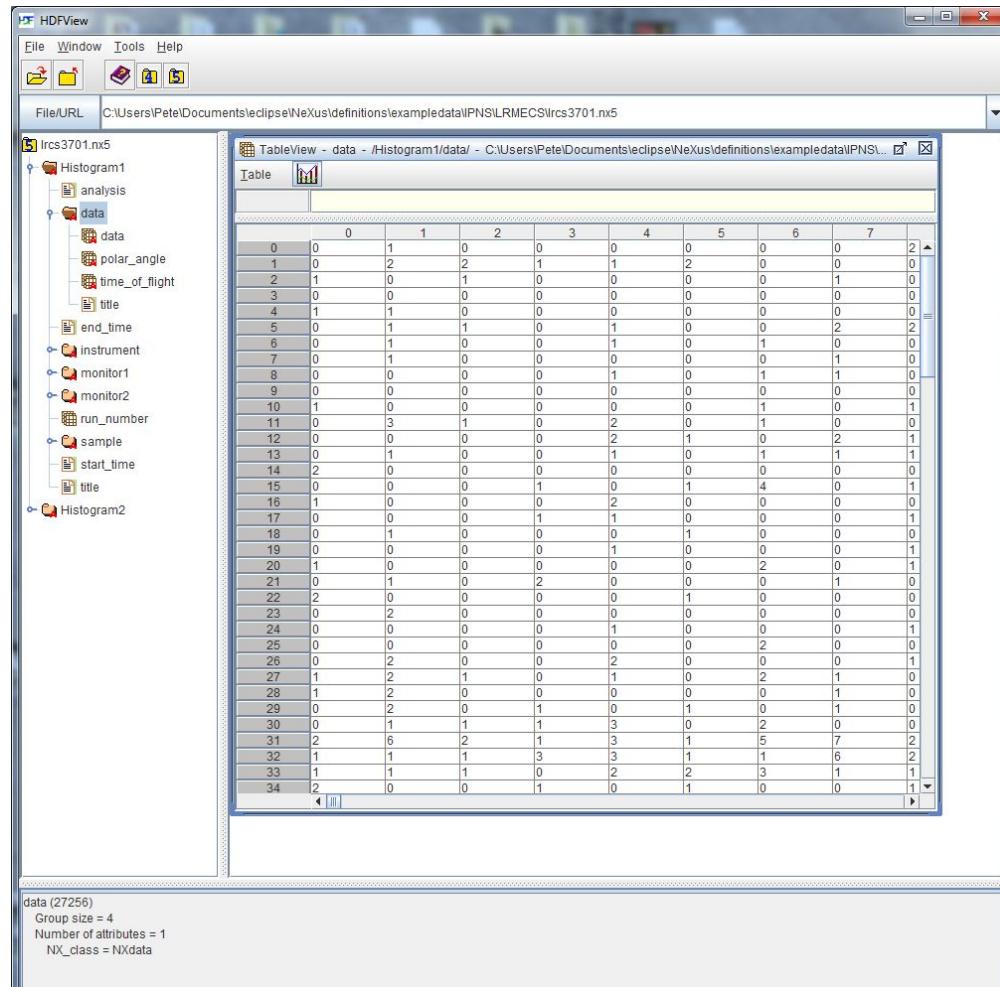
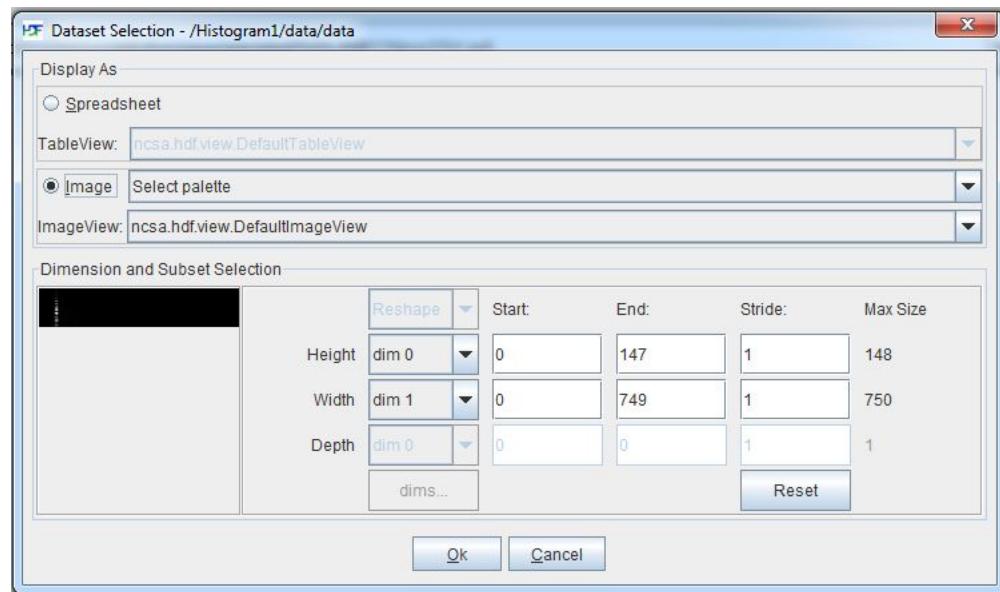
Visualize Using *HDFview*

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

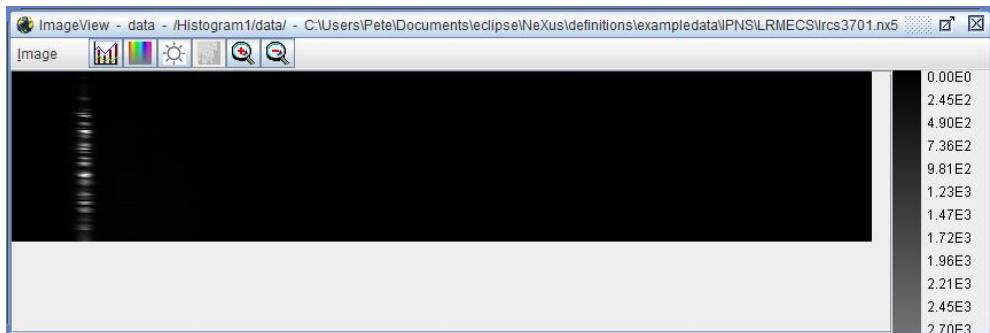


Fig. 9: LRMECS lrcs3701 data: *HDFview* Image

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.

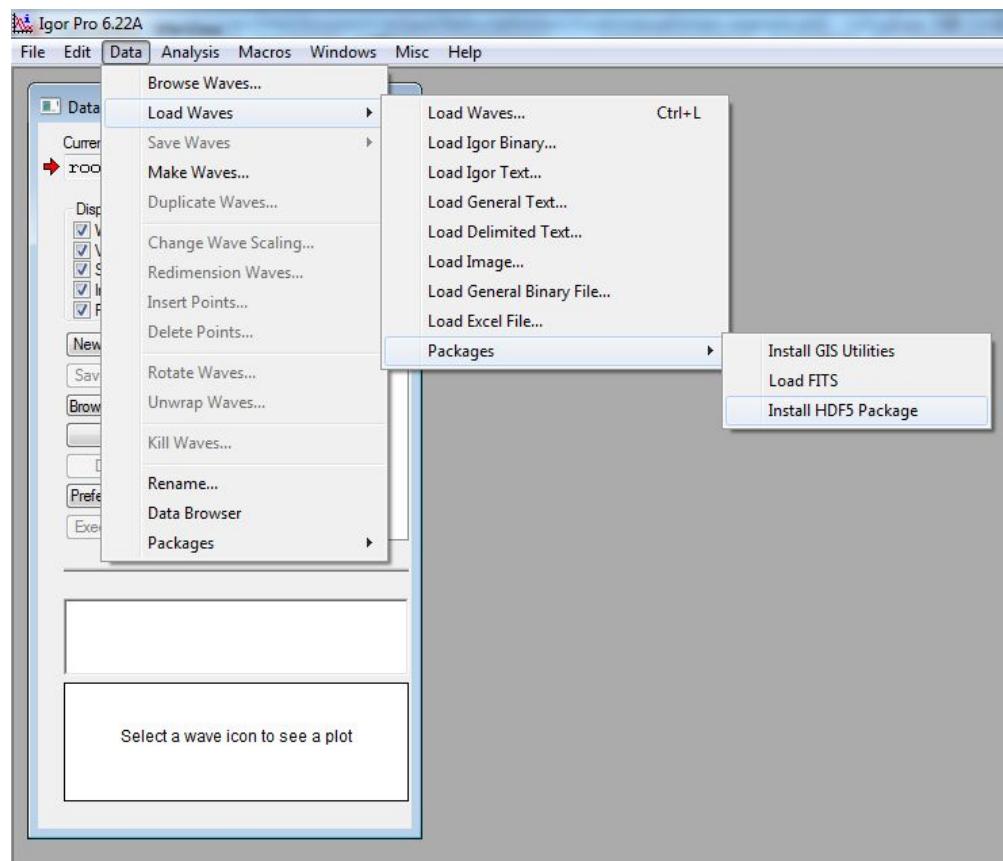


Fig. 10: LRMECS lrcs3701 data: *IgorPro* install HDF5 Browser

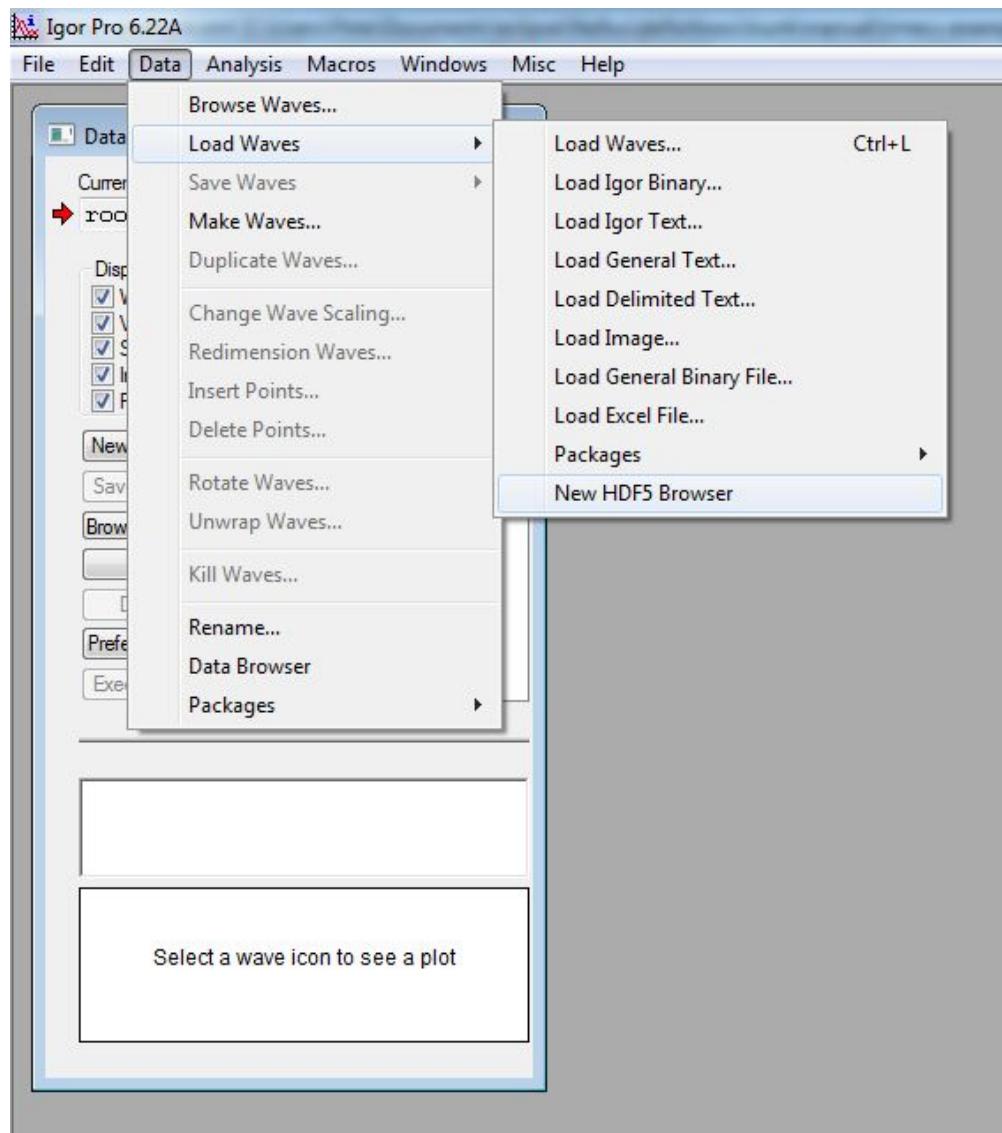
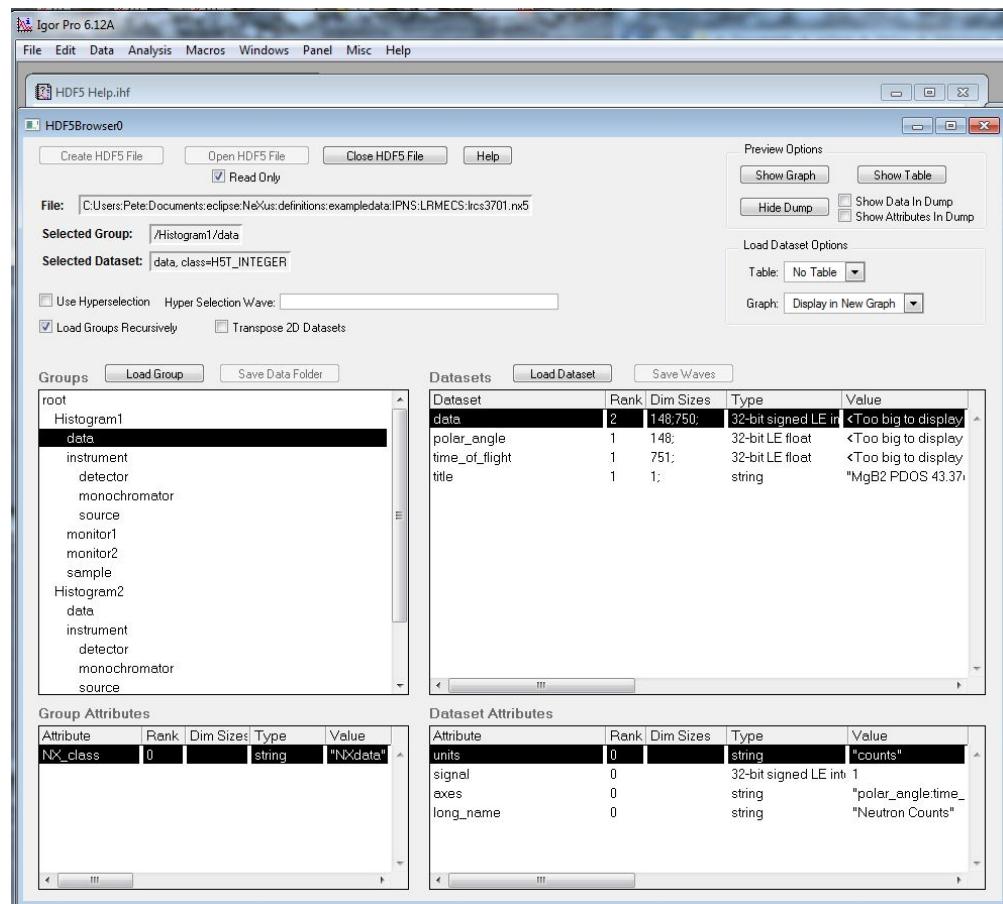
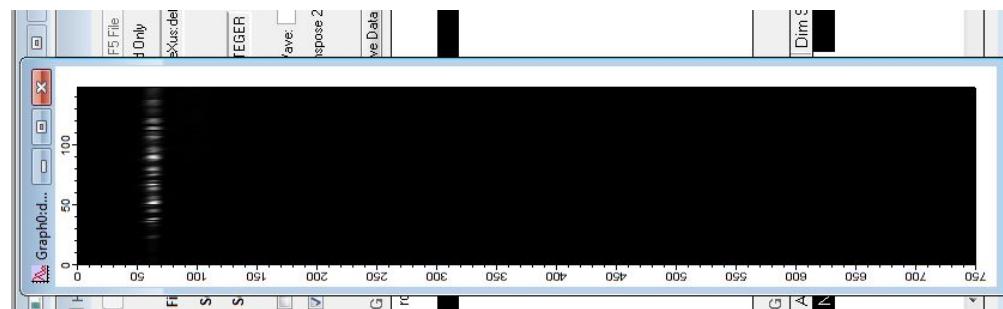


Fig. 11: LRMECS lrcs3701 data: *IgorPro HDFBrowser* dialog

Fig. 12: LRMECS lrcs3701 data: *IgorPro HDFBrowser* dialogFig. 13: LRMECS lrcs3701 data: *IgorPro Image*

HDF5 File Writing Plugin

This example describes how to write a NeXus HDF5 data file using the EPICS¹ Area Detector² HDF5 file writing plugin³. We will use the EPICS SimDetector⁴ 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
<code>attributes.xml</code>	what information to know about from EPICS and other sources
<code>layout.xml</code>	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.

```

1 <?xml version="1.0" standalone="no" ?>
2 <!-- Attributes -->
3 <Attributes
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:noNamespaceSchemaLocation=
6   "https://github.com/areaDetector/ADCore/blob/master/iocBoot/NDAttributes.xsd"
7 >
8   <Attribute name="AcquireTime"
9     type="EPICS_PV"
10    source="13SIM1:cam1:AcquireTime"
11    dbrtype="DBR_NATIVE"
12    description="Camera acquire time"/>
13   <Attribute name="ImageCounter"
14     type="PARAM"
```

(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)

```

15      source="ARRAY_COUNTER"
16      datatype="INT"
17      description="Image counter"/>
18  <Attribute name="calc1_val"
19      type="EPICS_PV"
20      source="$(P)userCalc1.VAL"
21      dbrtype="DBR_NATIVE"
22      description="some calculation result"/>
23  <Attribute name="calc2_val"
24      type="EPICS_PV"
25      source="$(P)userCalc2.VAL"
26      dbrtype="DBR_NATIVE"
27      description="another calculation result"/>
28  <Attribute name="MaxSizeX"
29      type="PARAM"
30      source="MAX_SIZE_X"
31      datatype="INT"
32      description="Detector X size"/>
33  <Attribute name="MaxSizeY"
34      type="PARAM"
35      source="MAX_SIZE_Y"
36      datatype="INT"
37      description="Detector Y size"/>
38  <Attribute name="CameraModel"
39      type="PARAM"
40      source="MODEL"
41      datatype="STRING"
42      description="Camera model"/>
43  <Attribute name="CameraManufacturer"
44      type="PARAM"
45      source="MANUFACTURER"
46      datatype="STRING"
47      description="Camera manufacturer"/>
48</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.

```

1  <?xml version="1.0" standalone="no" ?>
2  <hdf5_layout>
3    <group name="entry">

```

(continues on next page)

(continued from previous page)

```

4   <attribute name="NX_class" source="constant" value="NXentry" type="string"/>
5   <group name="instrument">
6     <attribute name="NX_class" source="constant" value="NXinstrument" type="string"/>
7     <group name="detector">
8       <attribute name="NX_class" source="constant" value="NXdetector" type="string"/>
9       <dataset name="data" source="detector" det_default="true">
10      <attribute name="NX_class" source="constant" value="SDS" type="string"/>
11      <attribute name="signal" source="constant" value="1" type="int"/>
12      <attribute name="target" source="constant" value="/entry/instrument/detector/
13        <data" type="string"/>
14      </dataset>
15      <group name="NDAttributes">
16        <attribute name="NX_class" source="constant" value="NXcollection" type="string
17        </>
18        <dataset name="ColorMode" source="ndattribute" ndattribute="ColorMode"/>
19      </group>          <!-- end group NDAttribute -->
20    </group>          <!-- end group detector -->
21    <group name="NDAttributes" ndattr_default="true">
22      <attribute name="NX_class" source="constant" value="NXcollection" type="string"/>
23      </>
24      </group>          <!-- end group NDAttribute (default) -->
25    <group name="performance">
26      <dataset name="timestamp" source="ndattribute"/>
27    </group>          <!-- end group performance -->
28  </group>          <!-- end group instrument -->
29  <group name="data">
30    <attribute name="NX_class" source="constant" value="NXdata" type="string"/>
31    <hardlink name="data" target="/entry/instrument/detector/data"/>
32    <!-- The "target" attribute in /entry/instrument/detector/data is used to
33      tell Nexus utilities that this is a hardlink -->
34  </group>          <!-- end group data -->
35  </group>          <!-- end group entry -->
36 </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:

```

<group
  name="NDAttributes"
  ndattr_default="true">
<attribute
  name="NX_class"
  source="constant"
  value="NXcollection"
  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 manual⁵ for placement advice if you are unsure.

⁵ NeXus manual: <https://manual.nexusformat.org/>

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 caQtDM⁶):

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
- 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”.

⁶ caQtDM: <http://epics.web.psi.ch/software/caqtdm/>

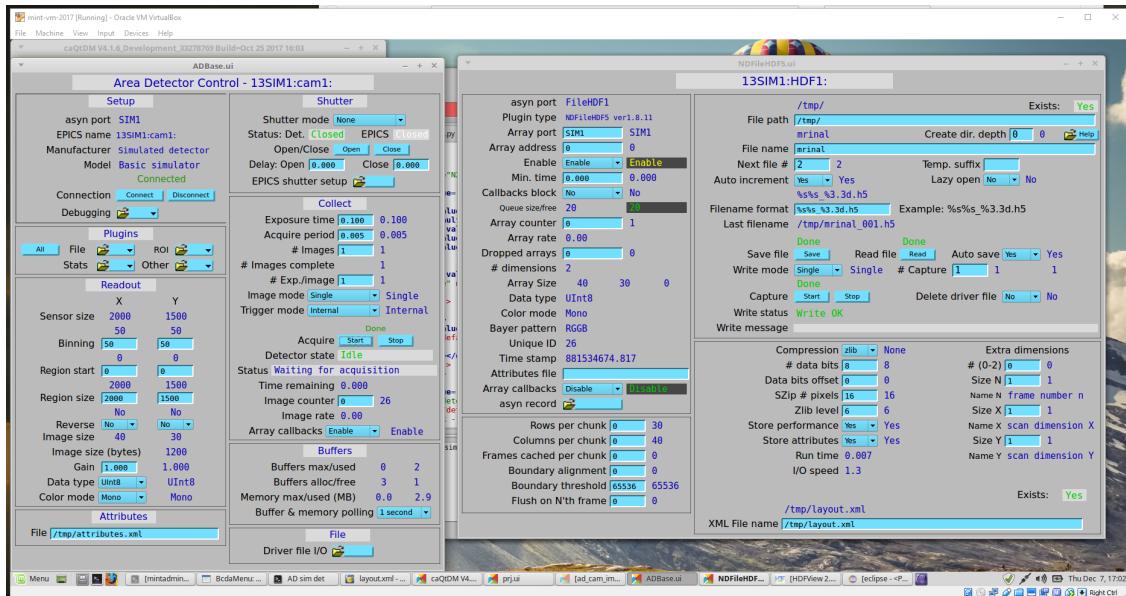


Fig. 14: ADbase and NDFileHDF5 configuration screens

- 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.

Validate the XML files

To validate an XML file, you’ll need the associated `.xsd` (XML Schema) file, then use this command:

```
xmllint --noout --schema path/to/local/schema/file.xsd file_to_validate.xml
```

These are the XML Schema files to use:

to validate use this XML Schema file
<code>attributes.xml</code>	<code>NDAtributes.xsd</code>
<code>layout.xml</code>	<code>hdf5_xml_layout_schema.xsd</code>
<code>nxd1.xsd</code>	<code>XMLSchema.xsd</code>
<code>nxd1Types.xsd</code>	<code>XMLSchema.xsd</code>
any NXDL file	<code>nxd1.xsd</code>

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.

```

1 import numpy as np
2 import h5py
3 import datetime
4
5 def write_nexus_file(fname, image, md={}):
6     """
7         write the image to a NeXus HDF5 data file
8
9     Parameters
10    -----
11     fname : str
12         name of the file (relative or absolute) to be written
13     image : numpy array
14         the image data
15     md : dictionary
16         key: value where value is something that can be written by h5py
17             (such as str, int, float, numpy array, ...)
18     """
19     nexus = h5py.File(fname, "w")
20     nexus.attrs["filename"] = fname
21     nexus.attrs["file_time"] = datetime.datetime.now().astimezone().isoformat()
22     nexus.attrs["creator"] = "write_nexus_file()"
23     nexus.attrs["H5PY_VERSION"] = h5py.__version__
24
25     # /entry
26     nxentry = nexus.create_group("entry")
27     nxentry.attrs["NX_class"] = "NXentry"
28     nexus.attrs["default"] = nxentry.name
29
30     # /entry/instrument
31     nxinstrument = nxentry.create_group("instrument")
32     nxinstrument.attrs["NX_class"] = "NXinstrument"
33
34     # /entry/instrument/detector
35     nxdetector = nxinstrument.create_group("detector")
36     nxdetector.attrs["NX_class"] = "NXdetector"
37
38     # /entry/instrument/detector/image
39     ds = nxdetector.create_dataset("image", data=image, compression="gzip")
40     ds.attrs["units"] = "counts"
41     ds.attrs["target"] = "/entry/instrument/detector/image"

```

(continues on next page)

⁷ h5py: <http://docs.h5py.org>

(continued from previous page)

```

42
43     # /entry/data
44     nxdata = nxentry.create_group("data")
45     nxdata.attrs["NX_class"] = "NXdata"
46     nxentry.attrs["default"] = nxdata.name
47
48     # /entry/data/data --> /entry/instrument/detector/image
49     nxdata["data"] = nexus["/entry/instrument/detector/image"]
50     nxdata.attrs["signal"] = "data"
51
52     if len(md) > 0:
53         # /entry/instrument/metadata (optional, for metadata)
54         metadata = nxinstrument.create_group("metadata")
55         metadata.attrs["NX_class"] = "NXcollection"
56         for k, v in md.items():
57             try:
58                 metadata.create_dataset(k, data=v)
59             except Exception:
60                 metadata.create_dataset(k, data=str(v))
61
62     nexus.close()
63
64
65 if __name__ == "__main__":
66     """demonstrate how to use this code"""
67     import epics
68     prefix = "13SIM1:"
69     img = epics.caget(prefix+"image1:ArrayData")
70     size_x = epics.caget(prefix+"cam1:ArraySizeX_RBV")
71     size_y = epics.caget(prefix+"cam1:ArraySizeY_RBV")
72     # edit the full image for just the binned data
73     img = img[:size_x*size_y].reshape((size_x, size_y))
74
75     extra_information = dict(
76         unique_id = epics.caget(prefix+"image1:UniqueId_RBV"),
77         size_x = size_x,
78         size_y = size_y,
79         detector_state = epics.caget(prefix+"cam1:DetectorState_RBV"),
80         bitcoin_value="15000",
81     )
82     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:

```

1 example.h5 : NeXus data file
2   @H5PY_VERSION = "3.6.0"
3   @creator = "write_nexus_file()"
4   @default = "entry"
5   @file_time = "2022-03-07 14:34:04.418733"
6   @filename = "example.h5"
7   entry:NXentry
8     @NX_class = "NXentry"

```

(continues on next page)

(continued from previous page)

```

9      @default = "data"
10     data:NXdata
11       @NX_class = "NXdata"
12       @signal = "data"
13       data --> /entry/instrument/detector/image
14   instrument:NXinstrument
15     @NX_class = "NXinstrument"
16   detector:NXdetector
17     @NX_class = "NXdetector"
18     image:NX_UINT8[1024,1024] = __array
19     __array = [
20       [76, 77, 78, '...', 75]
21       [77, 78, 79, '...', 76]
22       [78, 79, 80, '...', 77]
23       ...
24       [75, 76, 77, '...', 74]
25     ]
26     @target = "/entry/instrument/detector/image"
27     @units = "counts"
28   metadata:NXcollection
29     @NX_class = "NXcollection"
30     bitcoin_value:NX_CHAR = b'15000'
31     detector_state:NX_INT64[] =
32     size_x:NX_INT64[] =
33     size_y:NX_INT64[] =
34     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`:

```

1 import numpy as np
2 from nexusformat.nexus import *
3
4
5 def write_nexus_file(fname, image, md={}):
6     """
7         write the image to a NeXus HDF5 data file
8
9     Parameters

```

(continues on next page)

⁸ `nexusformat`: This Python package is described on the NeXPY web site

(continued from previous page)

```

10 -----
11     fname : str
12         name of the file (relative or absolute) to be written
13     image : numpy array
14         the image data
15     md : dictionary
16         key: value where value is something that can be written by h5py
17             (such as str, int, float, numpy array, ...)
18     .....
19
20     nx = NXroot()
21     nx['/entry'] = NXentry(NXinstrument(NXdetector()))
22     nx['entry/instrument/detector/image'] = NXfield(image, units='counts',
23                                                 compression='gzip')
24     nx['entry/data'] = NXdata()
25     nx['entry/data'].makelink(nx['entry/instrument/detector/image'])
26     nx['entry/data'].nxsignal = nx['entry/data/image']
27
28     if len(md) > 0:
29         # /entry/instrument/metadata (optional, for metadata)
30         metadata = nx['/entry/instrument/metadata'] = NXcollection()
31         for k, v in md.items():
32             metadata[k] = v
33
34
35     nx.save(fname, 'w')
36
37
38 if __name__ == "__main__":
39     """demonstrate how to use this code"""
40     import epics
41     prefix = "13SIM1:"
42     img = epics.caget(prefix+"image1:ArrayData")
43     size_x = epics.caget(prefix+"cam1:ArraySizeX_RBV")
44     size_y = epics.caget(prefix+"cam1:ArraySizeY_RBV")
45     # edit the full image for just the binned data
46     img = img[:size_x*size_y].reshape((size_x, size_y))
47
48     extra_information = dict(
49         unique_id = epics.caget(prefix+"image1:UniqueId_RBV"),
50         size_x = size_x,
51         size_y = size_y,
52         detector_state = epics.caget(prefix+"cam1:DetectorState_RBV"),
53         bitcoin_value="15000",
54     )
55     write_nexus_file("example.h5", img, md=extra_information)

```

Visualization

You can visualize the HDF5 files with several programs, such as: hdfview⁹, nexpy¹⁰, or pymca¹¹. Views of the test image shown using **NeXPy** (from the HDF5 file) and **caQtDM** (the image from EPICS) are shown.

⁹ hdfview: <https://support.hdfgroup.org/products/java/hdfview/>

¹⁰ nexpy: <https://nexpy.github.io/nexpy/>

¹¹ pymca: <http://pymca.sourceforge.net/>

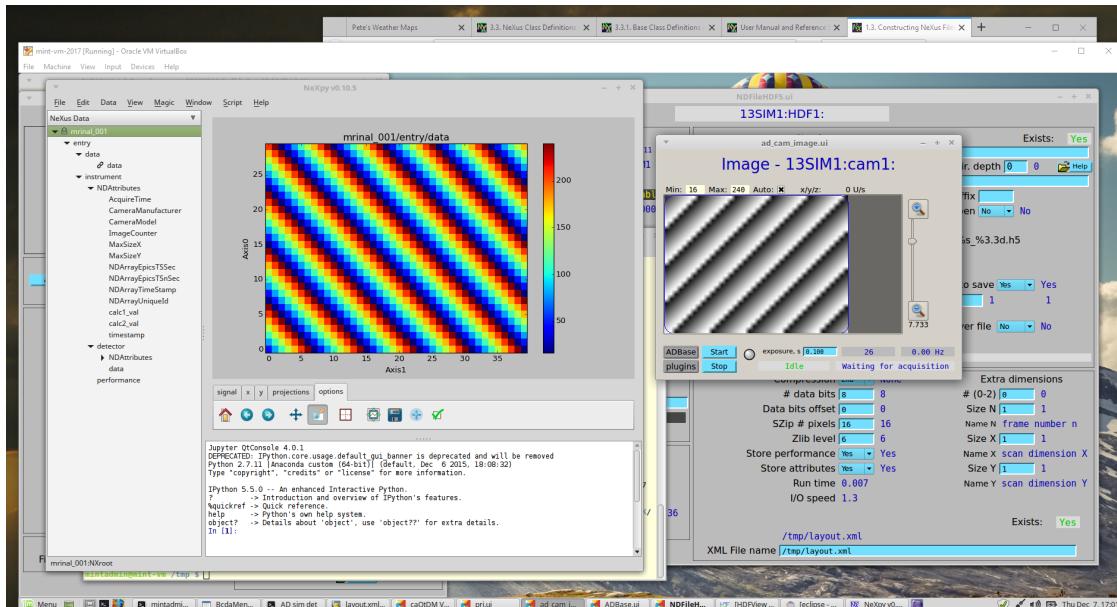


Fig. 15: Views of the image in NeXPY (left) and in caQtDM (right)

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	<code>write_nexus_file.py</code> rewritten with <code>nexusformat</code> package
example.h5	example NeXus HDF5 file written from Python

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.

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 GitHub and accessible via URL: <https://github.com/nexusformat/definitions/releases/tag/NXcrystal-1.0>

Description

General documentation if this NXDL file.

Symbols table

The *Symbols* table describes keywords used in this NXDL file to designate array dimensions. For reasons of avoiding naming collisions and to facilitate readability and comprehension for those whom are new to an NXDL file, the following guidelines are strongly encouraged:

- All symbols used in the application definition are defined in a single *Symbols* table.

- The *name* of a symbol uses camel case without any white space or underscores.

examples:

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](#)

```

1 <definition name="NXtomo" extends="NXobject" type="group"
2   category="application"
3   xmlns="http://definition.nexusformat.org/nxdl/3.1"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="http://definition.nexusformat.org/nxdl/3.1 ../nxdl.xsd"
6 >
7   <symbols>
8     <doc>
9       These symbols will be used below to coordinate datasets with the
10      ↵ same shape.
11     </doc>
12     <symbol name="nFrames">
13       <doc>Number of frames</doc>
14     </symbol>
15     <symbol name="xSize">
16       <doc>Number of pixels in X direction</doc>
17     </symbol>
18     <symbol name="ySize">
19       <doc>Number of pixels in Y direction</doc>
20     </symbol>
21   </symbols>
22   <doc>
23     This is the application definition for x-ray or neutron tomography raw
24     ↵ data.
25
26     In tomography
27     a number of dark field images are measured, some bright field images and,
28     ↵ of course the sample.
29     In order to distinguish between them images carry a image_key.
30     </doc>
31     <group type="NXentry" name="entry">
32       <field name="title" minOccurs="0" maxOccurs="1"/>
33     ...

```

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.

```

1  NameStartChar ::= _ | a..z | A..Z
2  NameChar      ::= NameStartChar | 0..9
3  Name          ::= NameStartChar (NameChar)*
4
5  Or, as a regular expression:    [_a-zA-Z] [_a-zA-Z0-9]*
6  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	Page 134, 1	unbounded
attribute	Page 134, 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`.

```

1  <choice name="pixel_shape">
2    <group type="NXoff_geometry">
3      <doc>
4        Shape description of each pixel. Use only if all pixels in the detector
5        are of uniform shape.
6      </doc>
7    </group>
8    <group type="NXcylindrical_geometry">
9      <doc>
10        Shape description of each pixel. Use only if all pixels in the detector
11        are of uniform shape and require being described by cylinders.
12      </doc>
13    </group>
14  </choice>
```

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

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).

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 programs¹ 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>`
```

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.

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

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

For more details, see: [attributeType](#)

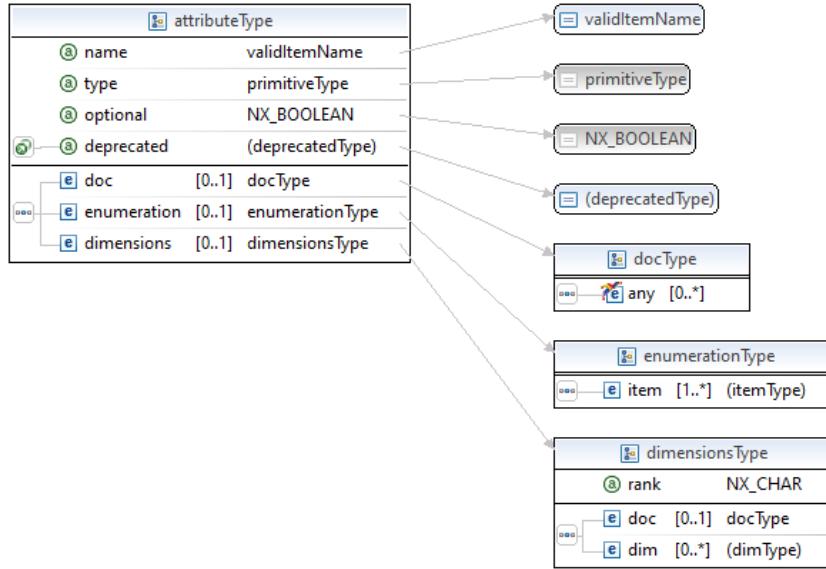


Fig. 1: Graphical representation of the NXDL attribute element

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](#)

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}:

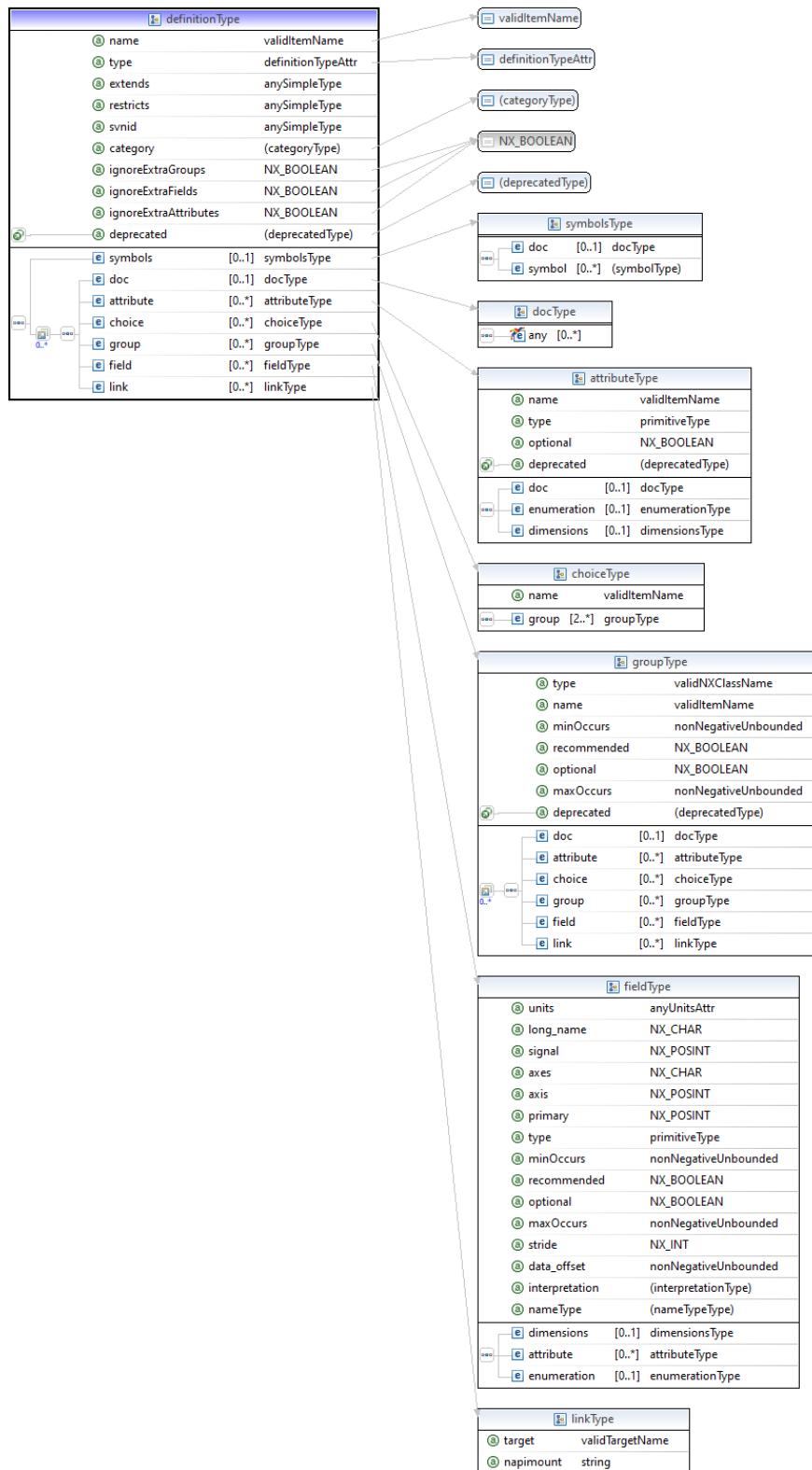


Fig. 2: Graphical representation of the NXDL definition element

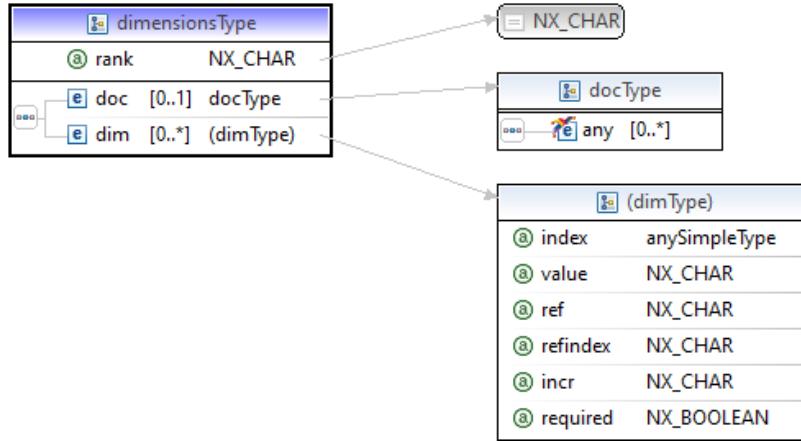


Fig. 3: Graphical representation of the NXDL dimensions element

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](#)

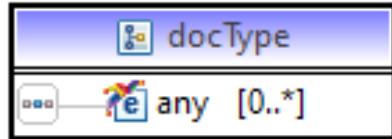


Fig. 4: Graphical representation of the NXDL doc element

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](#)

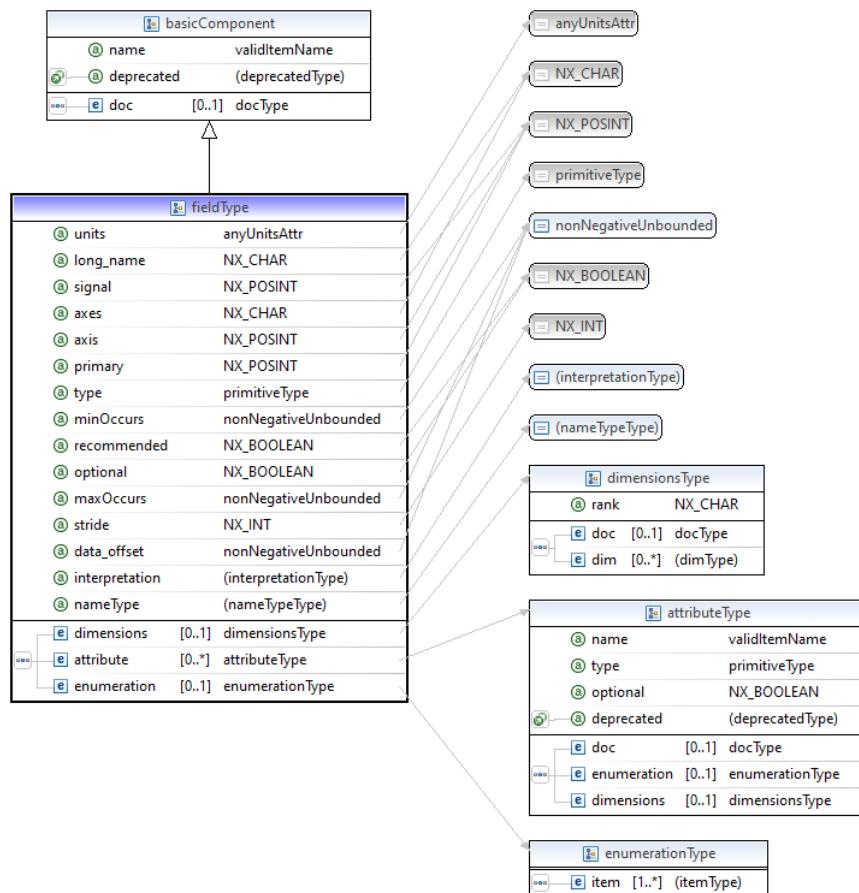
NXDL Field Types (internal)

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.)

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

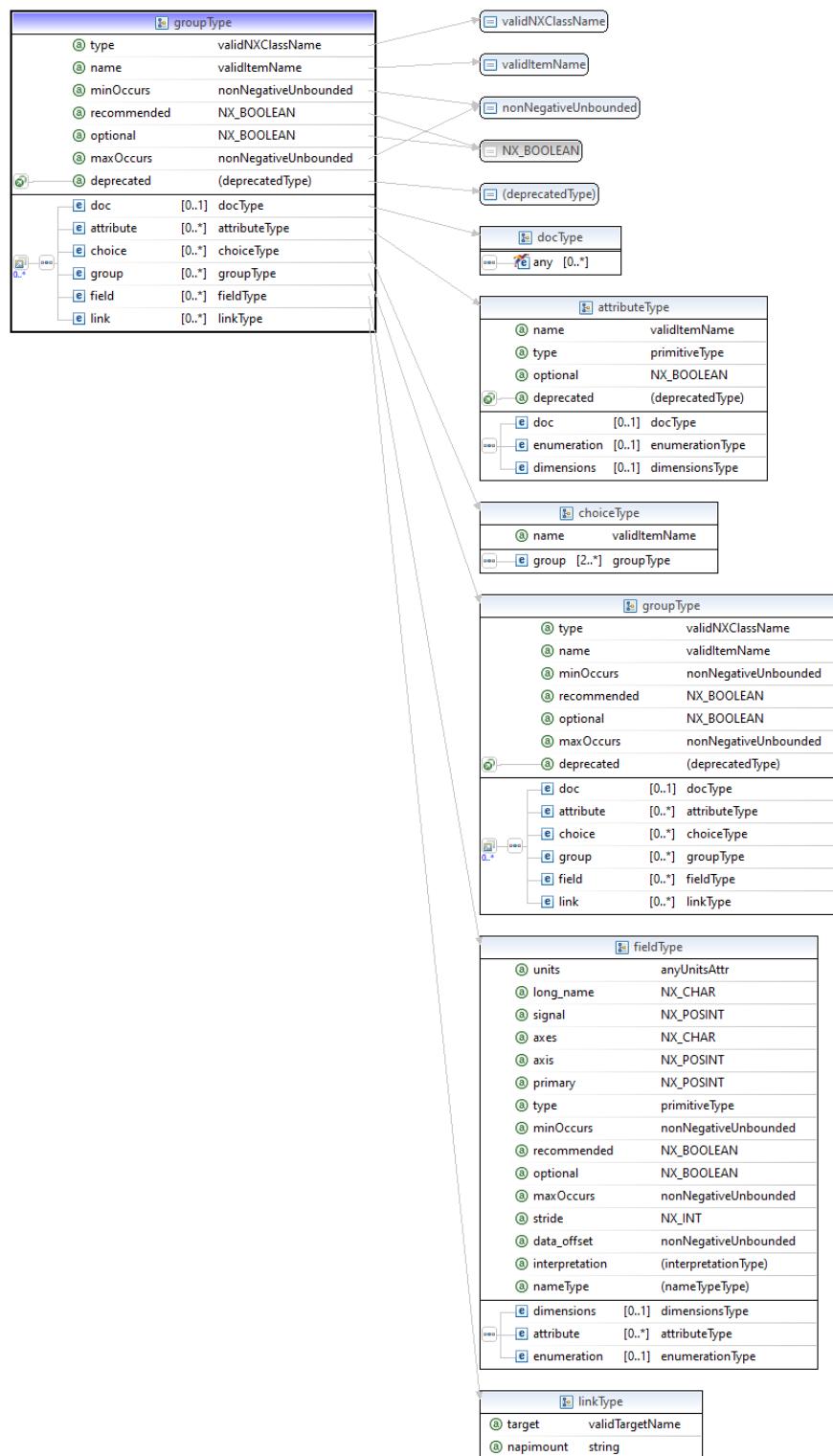


Fig. 7: Graphical representation of the NXDL group element

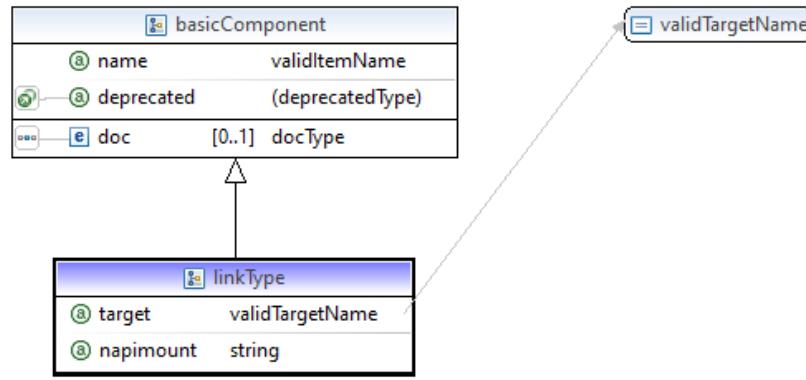


Fig. 8: Graphical representation of the NXDL link element

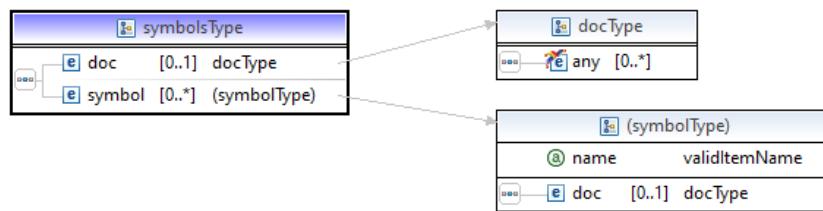


Fig. 9: Graphical representation of the NXDL symbols element

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 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 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 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`):

1

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

Enumerations can be closed or open. A closed enumeration is one where the list of values are the only ones allowed for a specification. An open enumeration is one where the list of values is incomplete and additional values other than those listed are allowed. Open enumerations should be used sparingly as the designer of the enumeration should try to find all possible values for a given field/attribute.

In case an open enumeration is used and the data provider wants to declare that the provided value deliberately not from the given list of values, the data provider should,

- for fields, set the attribute @custom=True.
- for attributes, set an additional attribute @my_attribute_custom=True, where my_attribute is the name of the attribute with the open enumeration.

Such attributes can be used deliberately to suppress warnings in NeXus validators.

Attributes of enumerationType

@open

Is this an open enumeration? An open enumeration allows additional values not listed.

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/phyperg.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 were 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.

@optional

A value of “true” is equivalent to minOccurs=0 while a value of “false” is equivalent to 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/phyperreg.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.

- If a unit is not provided by the list of NeXus unit categories, instead of providing a category, a field element can include an example of the units directly.
- The example does not constrain the scale of the units. For example, if the unit is eV/mm, the user could specify in a data file eV/cm, or any other unit that is convertible to the example given.

It is recommended that users and application developers check if their units and their unit examples adhere to the UDUNITS standard.¹

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.

¹ <https://www.unidata.ucar.edu/software/udunits/>

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 value of “true” is equivalent to minOccurs=0 while a value of “false” is equivalent to 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/analyser: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

A string of characters. The preferred string encoding is UTF-8. 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 table below.

When no unit category applies...

If a unit is not provided by the list of NeXus unit categories, instead of providing a category, a field element can include an example of the units directly.

The example does not constrain the scale of the units. For example, if the unit is eV/mm, the user could specify in a data file eV/cm, or any other unit that is convertible to the example given.

Example

```
<field name="monochromator_dispersion" units="eV/mm">
```

It is recommended that users and application developers check if their units and their unit examples adhere to the UDUNITS standard.¹

Conformance is not validated at this time.

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): m² | 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/cm²

NX_FREQUENCY

¹ <https://www.unidata.ucar.edu/software/udunits/>

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/cm³

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/length²,
example(s): 1/m²

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/m³

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

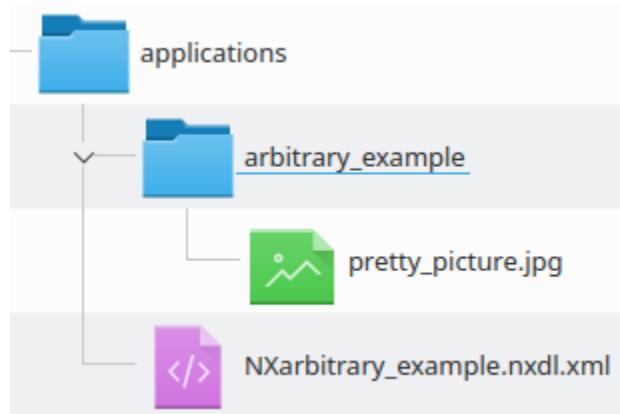
```
<definition name="NXarbitrary_example">

<!-- 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*) group¹ 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.

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.

NXaberration

Quantified aberration coefficient in an aberration_model.

NXactivity

A planned or unplanned action that has a temporal extension and for some time depends on some entity.

NXactuator

An actuator used to control an external condition.

NXaperture

A beamline aperture.

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.

NXatom

Base class for documenting a set of atoms.

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

NXcalibration

Subclass of NXprocess to describe post-processing calibrations.

NXcapillary

A capillary lens to focus the X-ray beam.

NXchemical_composition

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

NXcircuit

Base class for documenting circuit devices.

NXcite

A literature reference

NXcollection

An unvalidated set of terms, such as the description of a beam line.

NXcollimator

A beamline collimator.

NXcomponent

Base class for components of an instrument - real ones or simulated ones.

NXcoordinate_system

Base class to detail a coordinate system (CS).

NXcorrector_cs

Base class for a corrector reducing (spherical) aberrations in electron microscopy.

NXcrystal

A crystal monochromator or analyzer.

NXcs_computer

Base class for reporting the description of a computer

NXcs_memory

Base class for reporting the description of the memory system of a computer.

NXcs_process

Base class for reporting the description of processing units of a computer.

NXcs_profiling

Computer science description for performance and profiling data of an application.

NXcs_profiling_event

Computer science description of a profiling event.

NXcs_storage

Base class for reporting the description of the I/O of a computer.

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.

NXdeflector

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

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.

NXdistortion

Subclass of NXprocess to describe post-processing distortion correction.

NXbeam_column

Base class for a set of components providing a controllable electron beam.

NXelectron_detector

A subclass of NXdetector for detectors that detect electrons.

NXem_ebsd

Base class method-specific for Electron Backscatter Diffraction (EBSD).

NXem_eds

Base class method-specific for energy-dispersive X-ray spectroscopy (EDS/EDXS).

NXem_eels

Base class method-specific for Electron Energy Loss Spectroscopy (EELS).

NXem_img

Base class for method-specific generic imaging with electron microscopes.

NXem_measurement

Base class for documenting a measurement with an electron microscope.

NXem_simulation

Base class for documenting a simulation of electron beam-matter interaction.

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

NXevent_data_em

Base class to store state and (meta)data of events for electron microscopy.

NXfabrication

Details about a component as it is defined by its manufacturer.

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.

NXhistory

A set of activities that occurred to a physical entity prior/during experiment.

NXbeam_column

Base class for a set of components equipping an instrument with FIB capabilities.

NXimage

Base class for reporting a set of images representing specializations of NXdata.

NXinsertion_device

An insertion device, as used in a synchrotron light source.

NXinstrument

Collection of the components of the instrument or beamline.

NXinstrument_em

Base class for instrument-related details of a real or simulated electron microscope.

NXinteraction_volume_em

Base class to describe the volume of interaction for particle-matter interaction.

NXion

Base class for documenting the set of atoms of a (molecular) ion.

NXlens_em

Base class for an electro-magnetic lens or a compound lens.

NXlog

Information recorded as a function of time.

NXmanipulator

Base class to describe the use of manipulators and sample stages.

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. The groups and fields contained

NXoff_geometry

Geometry (shape) description.

NXoptical_system_em

Base class for qualifying an electron optical system.

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

NXpeak

Base class for describing a peak, its functional form, and support values

NXphase

Base class to describe a (thermodynamic) phase as a component of a material.

NXpid_controller

A description of a feedback system in terms of the settings of a proportional-integral-derivative (PID) controller.

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.

NXprogram

Base class to describe a software tool or library.

NXpump

Device to reduce an atmosphere (real or simulated) to a controlled pressure.

NXreflections

Reflection data from diffraction experiments

NXregistration

Describes image registration procedures.

NXresolution

Describes the resolution of a physical quantity.

NXroi

Base class to report on a region-of-interest of material analyzed or simulated.

NXroot

The root of a NeXus file.

NXrotations

Base class to detail a set of rotations, orientations, and disorientations.

NXsample

Any information on the sample.

NXsample_component

One group like this per component can be recorded for a sample consisting of multiple components.

NXscanbox_em

The scan box or scan controller is a component that is used to deflect a

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

Radiation source emitting a beam.

NXspectrum

Base class container for reporting a set of spectra.

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.

NXunit_cell

Base class to describe structural aspects of an arrangement of

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.

NXaberration

Status:

base class, extends [NXobject](#)

Description:

Quantified aberration coefficient in an aberration_model.

For an introduction in the aberrations in electron microscopy see R. Dunin-Borkowski et al. and 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. Conversion tables are also summarized by Y. Liao an introduction.

Symbols:

No symbol table

Groups cited:

none

Structure:

magnitude: (optional) *NX_NUMBER* {units=*NX_ANY*}

Magnitude of the aberration

magnitude_errors: (optional) *NX_NUMBER* {units=*NX_ANY*}

Uncertainty of the magnitude of the aberration

magnitude_errors_model: (optional) *NX_CHAR*

Free-text description how magnitude_errors was quantified e.g. via the 95% confidence interval, variance, standard deviation, using which algorithm or statistical model.

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

Time elapsed since the last measurement.

angle: (optional) *NX_NUMBER* {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*

Given name to this aberration.

alias: (optional) *NX_CHAR*

Alias to name or refer to this specific type of aberration.

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/magnitude_errors-field*](#)
- [*/NXaberration/magnitude_errors_model-field*](#)
- [*/NXaberration/name-field*](#)

NXDL Source:

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

NXactivity

Status:

base class, extends *NXObject*

Description:

A planned or unplanned action that has a temporal extension and for some time depends on some entity.

This class is a super class for all other activities.

Symbols:

No symbol table

Groups cited:

NXnote

Structure:

start_time: (optional) *NX_DATE_TIME*

Start time of this activity. It is recommended to include local time zone information.

end_time: (optional) *NX_DATE_TIME*

End time of this activity. It is recommended to include local time zone information.

@estimated: (optional) *NX_BOOLEAN*

In some cases, the end time of an activity can only be estimated. In this case, this attribute shall be True.

description: (optional) *NX_CHAR*

Short description of the activity.

NOTE: (optional) *NXnote*

This can be any data or other descriptor acquired during the activity (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.

Any number of instances of *NXnote* are allowed for describing extra details of this activity.

Hypertext Anchors

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

- */NXactivity/description-field*
- */NXactivity/end_time-field*
- */NXactivity/end_time@estimated-attribute*
- */NXactivity/NOTE-group*
- */NXactivity/start_time-field*

NXDL Source:

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

NXactuator**Status:**

base class, extends [NXobject](#)

Description:

An actuator used to control an external condition.

The condition itself is described in [NXenvironment](#).

Symbols:

No symbol table

Groups cited:

[NXfabrication](#), [NXpid_controller](#), [NXtransformations](#)

Structure:

name: (optional) [NX_CHAR](#)

Name of the actuator

short_name: (optional) [NX_CHAR](#)

Short name of actuator used e.g. on monitor display program

actuation_target: (optional) [NX_CHAR](#)

The physical component on which this actuator acts. This should be a path in the NeXus tree structure. For example, this could be an instance of NXsample or a device on NXinstrument.

physical_quantity: (optional) [NX_CHAR](#)

Name for the physical quantity effected by the actuation

Examples: temperature | pH | magnetic_field | electric_field | current | conductivity | resistance | voltage | pressure | flow | stress | strain | shear | surface_pressure

type: (optional) [NX_CHAR](#)

The type of hardware used for the actuation.

Examples (suggestions, but not restrictions):

Temperature

laser | gas lamp | filament | resistive

Pressure

anvil cell

Voltage

potentiostat

outputVALUE: (optional) [NX_NUMBER](#) {units=[NX_ANY](#)}

Any output that the actuator produces. For example, a heater can have the field output_power(NX_NUMBER).

depends_on: (optional) [NX_CHAR](#)

Refers to the last transformation specifying the position of the actuator in the NXtransformations chain.

PID_CONTROLLER: (optional) [NXpid_controller](#)

If the actuator is PID-controlled, the settings of the PID controller can be stored here.

TRANSFORMATIONS: (optional) *NXtransformations*

This is the group recommended for holding the chain of translation and rotation operations necessary to position the actuator within the instrument. The dependency chain may however traverse similar groups in other component groups.

FABRICATION: (optional) *NXfabrication***Hypertext Anchors**

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

- */NXactuator/actuation_target-field*
- */NXactuator/depends_on-field*
- */NXactuator/FABRICATION-group*
- */NXactuator/name-field*
- */NXactuator/outputVALUE-field*
- */NXactuator/physical_quantity-field*
- */NXactuator/PID_CONTROLLER-group*
- */NXactuator/short_name-field*
- */NXactuator/TRANSFORMATIONS-group*
- */NXactuator/type-field*

NXDL Source:

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

NXaperture**Status:**

base class, extends *NXcomponent*

Description:

A beamline aperture.

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

You can specify the geometry of the aperture using either NXoff_geometry or for simpler geometry shape and size.

Symbols:

No symbol table

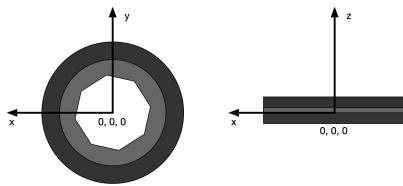
Groups cited:

NXgeometry, *NXnote*, *NXoff_geometry*, *NXpositioner*

Structure:**depends_on:** (optional) *NX_CHAR <=*

The reference point of the aperture is its center in the x and y axis. The reference point on the z axis is the surface of the aperture pointing towards the source.

In complex (asymmetric) geometries an NXoff_geometry group can be used to provide an unambiguous reference.



material: (optional) [NX_CHAR](#)

Absorbing material of the aperture

description: (optional) [NX_CHAR](#) <=

Description of aperture

shape: (optional) [NX_CHAR](#)

Shape of the aperture.

Any of these values:

- straight slit
- curved slit
- pinhole
- circle
- square
- hexagon
- octagon
- bladed
- open
- grid

size: (optional) [NX_NUMBER](#) {units=[NX_LENGTH](#)}

The relevant dimension for the aperture, i.e. slit width, pinhole and iris diameter

OFF_GEOMETRY: (optional) [NXoff_geometry](#)

Use this group to describe the shape of the aperture.

POSITIONER: (optional) [NXpositioner](#)

Stores the raw positions of aperture motors.

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/GEOOMETRY-group*](#)
- [*/NXaperture/material-field*](#)
- [*/NXaperture/NOTE-group*](#)
- [*/NXaperture/OFF_GEOOMETRY-group*](#)
- [*/NXaperture/POSITIONER-group*](#)
- [*/NXaperture/shape-field*](#)
- [*/NXaperture/size-field*](#)

NXDL Source:

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

NXatom

Status:

base class, extends *NXObject*

Description:

Base class for documenting a set of atoms.

Atoms in the set may be bonded. The set may have a net charge to represent an ion. An ion can be a molecular ion.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_pos: Number of atom positions.

d: Dimensionality

Groups cited:

none

Structure:

name: (optional) *NX_CHAR*

Given name for the set.

This field could for example be used in the research field of atom probe tomography to store a standardized human-readable name of the element or (molecular) ion like such as Al +++ or 12C +.

identifier_chemical: (optional) *NX_CHAR*

Identifier used to refer to if the set of atoms represents a substance.

Obligatory value: *inchi*

charge: (optional) *NX_NUMBER* {units=*NX_CHARGE*}

Signed net (partial) charge of the (molecular) ion.

Different methods for computing charge are in use. Care needs to be exercised with respect to the integration. [T. A. Manz](#) and [N. G. Limas](#) discuss computational details.

charge_state: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

Charge reported in multiples of the charge of an electron.

For research using atom probe tomography the value should be set to zero if the charge_state is unknown and irrecoverable. This can happen when classical ranging definition files in formats like RNG, RRNG are used. These file formats do not document the charge state explicitly but only the number of atoms of each element per molecular ion surplus the respective mass-to-charge-state-ratio interval.

Details on ranging definition files in the literature are [M. K. Miller](#).

volume: (optional) *NX_NUMBER* {units=*NX_VOLUME*}

Assumed volume affected by the set of atoms.

Neither individual atoms nor a set of cluster of these have a volume that is unique as a some cut-off criterion is required.

indices: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_pos])

Index for each atom at locations as detailed by position. Indices can be used as identifier and thus names for individual atoms.

type: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_pos]) {units=*NX_UNITLESS*}

Nuclide information for each atom at locations as detailed by position.

One [approach](#) for storing nuclide information efficiently is via hashing with the following formula

H is $H = Z + N \cdot 256$ with Z

the number of protons and N the number of neutrons of each nuclide given as 8-bit unsigned integer values.

position: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_pos, d]) {units=*NX_ANY*}

Position of each atom.

@reference_frame: (optional) *NX_CHAR*

Path to a reference frame in which positions are defined to resolve ambiguity when the reference frame is different to the NeXus default reference frame (McStas).

occupancy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_pos]) {units=*NX_DIMENSIONLESS*}

Relative occupancy of the atom position.

This field is useful for specifying the atomic motif in instances of *NXunit_cell*.

Hypertext Anchors

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

- */NXatom/charge-field*
- */NXatom/charge_state-field*
- */NXatom/identifier_chemical-field*
- */NXatom/indices-field*

- */NXatom/name-field*
- */NXatom/occupancy-field*
- */NXatom/position-field*
- */NXatom/position@reference_frame-attribute*
- */NXatom/type-field*
- */NXatom/volume-field*

NXDL Source:

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

NXattenuator**Status:**

base class, extends *NXcomponent*

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

Structure:

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

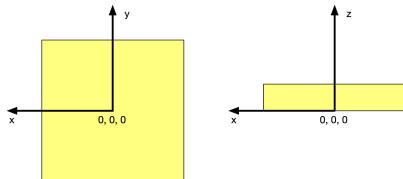
@time: (optional) *NX_DATE_TIME*

time stamp for this observation

depends_on: (optional) *NX_CHAR* <=

The reference point of the attenuator is its center in the x and y axis. The reference point on the z axis is the surface of the attenuator pointing towards the source.

In complex (asymmetric) geometries an NXoff_geometry group can be used to provide an unambiguous reference.

**shape:** (optional) *NXoff_geometry*

Shape of this component. Particularly useful to define the origin for position and orientation in non-standard cases.

Hypertext Anchors

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

- [*/NXattenuator/absorption_cross_section-field*](#)
- [*/NXattenuator/attenuator_transmission-field*](#)
- [*/NXattenuator/depends_on-field*](#)
- [*/NXattenuator/distance-field*](#)
- [*/NXattenuator/scattering_cross_section-field*](#)
- [*/NXattenuator/shape-group*](#)
- [*/NXattenuator/status-field*](#)
- [*/NXattenuator/status@time-attribute*](#)
- [*/NXattenuator/thickness-field*](#)
- [*/NXattenuator/type-field*](#)

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`, `incident_energy`, and related fields can be a scalar values or arrays, depending on the use case. To support these use cases, the explicit dimensionality of these fields is not specified, but it can be inferred by the presence of and shape of accompanying fields, such as `incident_wavelength_weights` for a polychromatic beam.

Symbols:

These symbols coordinate datasets with the same shape.

nP: Number of scan points.

m: Number of channels in the incident beam spectrum, if known

c: Number of moments representing beam divergence (x, y, xy, etc.)

Groups cited:

[NXdata](#), [NXtransformations](#)

Structure:

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 given location.

Several use cases are permitted, depending on the presence or absence of other `incident_energy_X` fields. The usage should follow that of `incident_wavelength`.

incident_energy_spread: (optional) [NX_NUMBER](#) {units=[NX_ENERGY](#)}

The energy spread FWHM for the corresponding energy(ies) in `incident_energy`. The usage of this field should follow that of `incident_wavelength`.

incident_energy_weights: (optional) [NX_NUMBER](#) {units=[NX_ENERGY](#)}

Relative weights of the corresponding energies in `incident_energy`. The usage of this field should follow that of `incident_wavelength`.

final_energy: (optional) [NX_FLOAT](#) (Rank: 1, Dimensions: [m]) {units=[NX_ENERGY](#)}

Energy carried by each particle of the beam on leaving the given location

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. If applicable, the first dimension shall represent the extent in the direction parallel to the azimuthal reference plane (by default it is [1,0,0]), and the second dimension shall be the normal to the reference plane (by default it is [0,1,0]).

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

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

Energy of a single pulse at the given location.

average_power: (optional) *NX_FLOAT* {units=*NX_POWER*}

Average power at the at the given location.

fluence: (optional) *NX_FLOAT* {units=mJ/cm²}

Incident energy fluence at the given location.

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

FWHM duration of the pulses at the given location.

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

Delay time between two pulses of a pulsed beam.

@reference_beam: (optional) *NX_CHAR*

A reference to the beam in relation to which the delay is.

This should be the path to another instance of NXbeam. The use of this attribute should be similar to that of the *depends_on attribute*. in NXtransformations.

frog_trace: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [nx, ny])

FROG (frequency-resolved optical gating) trace of the pulse.

This is to be used for ultrashort laser pulses in a FROG (frequency-resolved optical gating) setup.

frog_delays: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [nx]) {units=*NX_TIME*}

Horizontal axis of a FROG trace, i.e. delay.

This is to be used for ultrashort laser pulses in a FROG (frequency-resolved optical gating) setup.

frog_frequencies: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [ny]) {units=*NX_FREQUENCY*}

Vertical axis of a FROG trace, i.e. frequency.

This is to be used for ultrashort laser pulses in a FROG (frequency-resolved optical gating) setup.

chirp_type: (optional) *NX_CHAR*

The type of chirp implemented

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

Group delay dispersion of the pulse for linear chirp

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.

BEAMdirection: (optional) *NX_NUMBER* {units=*NX_UNITLESS*} <=

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. Note, this field should be referenced by the parent group's depends_on field; also, as this field is a direction, its transformation_type attribute should be omitted.

@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_UNITLESS*} <=

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]. Note, as this field is a direction, its `transformation_type` attribute should be omitted.

@vector: (optional) `NX_NUMBER` <=

Three values that define the direction of reference plane normal

@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/average_power-field`](#)
- [`/NXbeam/chirp_GDD-field`](#)
- [`/NXbeam/chirp_type-field`](#)
- [`/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/fluence-field`](#)
- [`/NXbeam/flux-field`](#)
- [`/NXbeam/frog_delays-field`](#)
- [`/NXbeam/frog_frequencies-field`](#)
- [`/NXbeam/frog_trace-field`](#)
- [`/NXbeam/incident_beam_divergence-field`](#)
- [`/NXbeam/incident_energy-field`](#)
- [`/NXbeam/incident_energy_spread-field`](#)
- [`/NXbeam/incident_energy_weights-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/pulse_delay-field*
- */NXbeam/pulse_delay@reference_beam-attribute*
- */NXbeam/pulse_duration-field*
- */NXbeam/pulse_energy-field*
- */NXbeam/TRANSFORMATIONS-group*
- */NXbeam/TRANSFORMATIONS/BEAMdirection-field*
- */NXbeam/TRANSFORMATIONS/BEAMdirection@depends_on-attribute*
- */NXbeam/TRANSFORMATIONS/BEAMdirection@offset-attribute*
- */NXbeam/TRANSFORMATIONS/BEAMdirection@vector-attribute*
- */NXbeam/TRANSFORMATIONS/reference_plane-field*
- */NXbeam/TRANSFORMATIONS/reference_plane@depends_on-attribute*
- */NXbeam/TRANSFORMATIONS/reference_plane@vector-attribute*

NXDL Source:

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

NXbeam_stop**Status:**

base class, extends *NXcomponent*

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*

Structure:

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*}

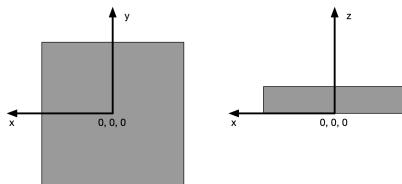
distance of the beamstop to the detector. Note, it is recommended to use NXtransformations instead.

status: (optional) *NX_CHAR*

Any of these values: in | out

depends_on: (optional) *NX_CHAR* <=

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.

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/x-field*
- */NXbeam_stop/y-field*

NXDL Source:

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

NXbending_magnet**Status:**

base class, extends [NXcomponent](#)

Description:

A bending magnet

Symbols:

No symbol table

Groups cited:

[NXdata](#), [NXgeometry](#), [NXoff_geometry](#)

Structure:

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](#) <=

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

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*

NXDL Source:

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

NXcalibration

Status:

base class, extends *NXprocess*

Description:

Subclass of NXprocess to describe post-processing calibrations.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays

ncal: Number of points of the calibrated and uncalibrated axes

Groups cited:

NXdata, *NXnote*, *NXparameters*

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 the procedures employed.

physical_quantity: (optional) *NX_CHAR*

The physical quantity of the calibration, e.g., energy, momentum, time, etc.

identifier_calibration_method: (optional) *NX_CHAR*

A digital persistent identifier (e.g., DOI, ISO standard) referring to a detailed description of a calibration method but no actual calibration data.

identifier_calibration_reference: (optional) *NX_CHAR*

A digital persistent identifier (e.g., a DOI) referring to a publicly available calibration measurement used for this instrument, e.g., a measurement of a known standard containing calibration information. The axis values may be copied or linked in the appropriate NXcalibration fields for reference.

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?

original_axis: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [ncal]) {units=*NX_ANY*}

Array containing the data coordinates in the original uncalibrated axis

@symbol: (optional) *NX_CHAR*

The symbol of the axis to be used in the fit_function, e.g., *energy*, *E*. This should comply to the following naming rules (similar to python's naming rules):

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

@input_path: (optional) *NX_CHAR*

The path from which this data is derived, e.g., raw detector axis. Should be a valid NeXus path name, e.g., /entry/instrument/detector/raw.

fit_formula_description: (optional) *NX_CHAR*

For non-linear energy calibrations. Here we can store a description of the formula used for the fit function.

Use a₀, a₁, ..., a_n for the coefficients, corresponding to the values in the coefficients group.

Use x₀, x₁, ..., x_m for the nth position in the *original_axis* field. If there is the symbol attribute specified for the *original_axis* this may be used instead of x. If you want to use the whole axis use x. Alternate axis can also be available as specified by the *input_SYMBOL* group. The data

should then be referred here by the *SYMBOL* name, e.g., for a field name *input_my_field* it should be referred here by *my_field* or *my_field0* if you want to read the zeroth element of the array.

scaling_factor: (optional) *NX_FLOAT* {units=*NX_ANY*}

For linear calibration. Scaling parameter. This should yield the relation *calibrated_axis* = (*original_axis* + *offset*) * *scaling_factor*.

For a more detailed description of scaling factors, see */NXdata/FIELDNAME_scaling_factor*.

offset: (optional) *NX_FLOAT* {units=*NX_ANY*}

For linear calibration. Offset parameter. This should yield the relation *calibrated_axis* = (*original_axis* + *offset*) * *scaling_factor*.

For a more detailed description of offset, see */NXdata/FIELDNAME_offset*.

mapping_MAPPING: (optional) *NX_FLOAT*

Mapping data for calibration.

This can be used to map data points from uncalibrated to calibrated values, i.e., by multiplying each point in the input axis by the corresponding point in the mapping data.

calibrated_axis: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [ncal]) {units=*NX_ANY*}

An array representing the axis after calibration, matching the data length

calibration_object: (optional) *NXnote* <=

A file serialisation of a calibration which may not be publicly available (externally from the NeXus file).

This metadata can be a documentation of the source (file) or database (entry) from which pieces of information have been extracted for consumption (e.g. in a research data management system (RDMS)). It is also possible to include the actual file by using the *file* field.

The axis values may be copied or linked in the appropriate NXcalibration fields for reference.

fit_formula_inputs: (optional) *NXparameters*

Additional input axis to be used in the formula.

TERM: (optional) *NX_CHAR* <=

The name of each TERM is used as the symbol to be used in the fit_formula_description, i.e., if the field name is *my_field* you should refer to this axis by *my_field* in the fit_formula_description.

@input_path: (optional) *NX_CHAR*

The path from which this data is derived, e.g., raw detector axis. Should be a valid NeXus path name, e.g., /entry/instrument/detector/raw.

coefficients: (optional) *NXparameters*

For non-linear energy calibrations, e.g. in a time-of-flight (TOF) detector, a polynomial function is fitted to a set of features (peaks) at well defined energy positions to determine E(TOF). Here we can store the fit coefficients.

aN: (optional) *NX_CHAR* <=

Use *a0*, *a1*, ..., *an* for the coefficients, corresponding to the values in the fit_formula_description.

DATA: (optional) *NXdata*

Any data acquired/used during the calibration that does not fit the *NX_FLOAT* fields above.
 NXdata groups can be used for multidimensional data which are relevant to the calibration

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/calibration_object-group*](#)
- [*/NXcalibration/coefficients-group*](#)
- [*/NXcalibration/coefficients/aN-field*](#)
- [*/NXcalibration/DATA-group*](#)
- [*/NXcalibration/description-field*](#)
- [*/NXcalibration/fit_formula_description-field*](#)
- [*/NXcalibration/fit_formula_inputs-group*](#)
- [*/NXcalibration/fit_formula_inputs/TERM-field*](#)
- [*/NXcalibration/fit_formula_inputs/TERM@input_path-attribute*](#)
- [*/NXcalibration/identifier_calibration_method-field*](#)
- [*/NXcalibration/identifier_calibration_reference-field*](#)
- [*/NXcalibration/last_process-field*](#)
- [*/NXcalibration/mapping_MAPPING-field*](#)
- [*/NXcalibration/offset-field*](#)
- [*/NXcalibration/original_axis-field*](#)
- [*/NXcalibration/original_axis@input_path-attribute*](#)
- [*/NXcalibration/original_axis@symbol-attribute*](#)
- [*/NXcalibration/physical_quantity-field*](#)
- [*/NXcalibration/scaling_factor-field*](#)
- [*/NXcalibration@default-attribute*](#)

NXDL Source:

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

NXcapillary

Status:

base class, extends *NXcomponent*

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***Structure:****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* <=**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

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/transmission-group*
- */NXcapillary/type-field*
- */NXcapillary/working_distance-field*

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

[NXatom](#)

Structure:

normalization: (optional) [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 which are of a particular type. By contrast, weight_percent normalization factorizes in the respective mass of the elements. Software libraries that work with units (like pint in Python) are challenged by these differences as at.-% and wt.-% are both fractional quantities.

Any of these values: `atom_percent | weight_percent`

total: (optional) [NX_NUMBER](#) (Rank: 1, Dimensions: [n]) {units=[NX_UNITLESS](#)}

Total based on which composition information is normalized.

ELEMENT: (optional) [NXatom](#)

Instances names for groups should use the chemical symbol of the element as it is defined in the periodic table.

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_FLOAT](#) {units=[NX_DIMENSIONLESS](#)}

Composition value for the element/ion referred to under name. Composition is reported either normalized for atomic or weight percent. The field normalization should be used to communicate this explicitly. Composition should be reported consistently for all instances ELEMENT.

composition_errors: (recommended) [NX_FLOAT](#) {units=[NX_DIMENSIONLESS](#)}

Magnitude of the standard deviation of the composition.

Hypertext Anchors

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

- [/NXchemical_composition/ELEMENT-group](#)
- [/NXchemical_composition/ELEMENT/composition-field](#)
- [/NXchemical_composition/ELEMENT/composition_errors-field](#)

- */NXchemical_composition/ELEMENT/count-field*
- */NXchemical_composition/normalization-field*
- */NXchemical_composition/total-field*

NXDL Source:

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

NXcircuit**Status:**

base class, extends *NXcomponent*

Description:

Base class for documenting circuit devices.

Electronic circuits are hardware components that connect several electronic components to achieve specific functionality, e.g. amplifying a voltage or convert a voltage to binary numbers, etc.

Symbols:

Constant to be used in the definition: the number of channels of the circuit board.

N_channel: Number of channels of the circuit board.

Groups cited:

NXcalibration, *NXfabrication*

Structure:

components: (optional) *NX_CHAR*

List of components used in the circuit, e.g., resistors, capacitors, transistors or any other complex components.

connections: (optional) *NX_CHAR*

Description of how components are interconnected, including connection points and wiring.

power_source: (optional) *NX_CHAR*

Details of the power source for the circuit, including voltage and current ratings.

signal_type: (optional) *NX_CHAR*

Type of signal (input signal) the circuit is designed to handle, e.g., analog, digital, mixed-signal.

operating_frequency: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The operating frequency of the circuit, see also bandwidth, which is possibly but not necessarily centered around this frequency (e.g. running a 100 kHz bandwidth amplifier at low, audio frequencies 1 - 20,000 Hz).

bandwidth: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The bandwidth of the frequency response of the circuit.

input_impedance: (optional) *NX_NUMBER* {units=*NX_ANY*}

Input impedance of the circuit.

output_impedance: (optional) *NX_NUMBER* {units=*NX_ANY*}

Output impedance of the circuit.

gain: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

Gain of the circuit, if applicable, usually all instruments have a gain which might be important or not.

noise_level: (optional) *NX_NUMBER* {units=*NX_ANY*}

Root-mean-square (RMS) noise level (in current or voltage) in the circuit in voltage or current.

temperature_range: (optional) *NX_NUMBER* {units=*NX_ANY*}

Operating temperature range of the circuit.

offset: (optional) *NX_NUMBER* {units=*NX_ANY*}

Offset value for current or voltage.

output_channels: (optional) *NX_NUMBER*

Number of output channels collected to this circuit. Most probably N_channel.

output_signal: (optional) *NX_NUMBER* {units=*NX_ANY*}

Type of output signal, e.g., voltage, current, digital.

power_consumption: (optional) *NX_NUMBER* {units=*NX_ANY*}

Power consumption of the circuit per unit time.

status_indicators: (optional) *NX_CHAR*

Status indicators for the circuit, e.g., LEDs, display readouts.

protection_features: (optional) *NX_CHAR*

Protection features built into the circuit, e.g., overvoltage protection, thermal shutdown.

acquisition_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

Updated rate for several processes using the input signal, e.g., History Graph, the circuit uses for any such process.

output_slew_rate: (optional) *NX_CHAR*

The rate at which the signal changes when ramping from the starting value.

hardware: (optional) *NXfabrication <=*

Hardware where the circuit is implanted; includes information about the hardware manufacturers and type (e.g. part number) All the elements below may be single numbers of an array of values with length N_channel describing multiple input and output channels.

calibration: (optional) *NXcalibration*

Calibration data for the circuit.

Hypertext Anchors

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

- */NXcircuit/acquisition_time-field*
- */NXcircuit/bandwidth-field*
- */NXcircuit/calibration-group*
- */NXcircuit/components-field*
- */NXcircuit/connections-field*
- */NXcircuit/gain-field*

- */NXcircuit/hardware-group*
- */NXcircuit/input_impedance-field*
- */NXcircuit/noise_level-field*
- */NXcircuit/offset-field*
- */NXcircuit/operating_frequency-field*
- */NXcircuit/output_channels-field*
- */NXcircuit/output_impedance-field*
- */NXcircuit/output_signal-field*
- */NXcircuit/output_slew_rate-field*
- */NXcircuit/power_consumption-field*
- */NXcircuit/power_source-field*
- */NXcircuit/protection_features-field*
- */NXcircuit/signal_type-field*
- */NXcircuit/status_indicators-field*
- */NXcircuit/temperature_range-field*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/base_classes/NXcircuit.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:

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*

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.

NXcollection content is not validated.

NXcollection is and will always be for unvalidated content.

Any and all content within a *NXcollection* group specified by an application definition cannot be validated.

It is suggested to use a *NXparameters* group for similar content which should 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 [NXcomponent](#)

Description:

A beamline collimator.

Symbols:

No symbol table

Groups cited:

[NXgeometry](#), [NXlog](#), [NXoff_geometry](#)

Structure:

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](#)

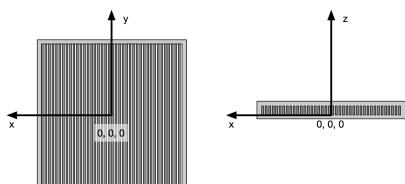
name of absorbing material

transmitting_material: (optional) [NX_CHAR](#)

name of transmitting material

depends_on: (optional) [NX_CHAR](#) <=

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

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/transmitting_material-field*
- */NXcollimator/type-field*

NXDL Source:

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

NXcomponent

Status:

base class, extends *NXObject*

Description:

Base class for components of an instrument - real ones or simulated ones.

Symbols:

No symbol table

Groups cited:

NXfabrication, *NXprogram*, *NXtransformations*

Structure:

applied: (optional) *NX_BOOLEAN*

Was the component used?

name: (optional) *NX_CHAR*

Name of the component.

description: (optional) *NX_CHAR*

Ideally, use instances of `identifierNAME` to point to a resource that provides further details.

If such a resource does not exist or should not be used, use this free text, although it is not recommended.

inputs: (optional) *NX_CHAR*

Instance or list of instances of `NXcomponent` (or base classes extending `NXcomponent`) or `NXbeam` that act as input(s) to this component.

Each input should point to the path of the group acting as input.

An example usage would be to chain components and/or beams together to describe the beam path in an experiment.

outputs: (optional) *NX_CHAR*

Instance or list of instances of `NXcomponent` (or base classes extending `NXcomponent`) or `NXbeam` that act as output(s) of this component.

For more information, see *inputs*.

depends_on: (optional) *NX_CHAR*

Specifies the position of the component by pointing to the last transformation in the transformation chain that is defined via the `NXtransformations` group.

NeXus positions components by applying a set of translations and rotations 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. The `depends_on` can also point to an instance of `NX_coordinate_system` if the transformations of the component depend on that coordinate system.

Usually these operations are stored in a `NXtransformations` group. But NeXus allows them to be stored anywhere.

FABRICATION: (optional) *NXfabrication*

PROGRAM: (optional) *NXprogram*

TRANSFORMATIONS: (optional) *NXtransformations*

Collection of axis-based translations and rotations to describe the location and geometry of the component in 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.

- */NXcomponent/applied-field*
- */NXcomponent/depends_on-field*
- */NXcomponent/description-field*
- */NXcomponent/FABRICATION-group*

- */NXcomponent/inputs-field*
- */NXcomponent/name-field*
- */NXcomponent/outputs-field*
- */NXcomponent/PROGRAM-group*
- */NXcomponent/TRANSFORMATIONS-group*

NXDL Source:

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

NXcoordinate_system**Status:**

base class, extends *NXObject*

Description:

Base class to detail a coordinate system (CS).

Instances of `NXcoordinate_system` can be used to describe coordinate systems other than the default NeXus coordinate system.

Whenever possible, all instances of `NXcoordinate_system` should be used at the top-level (i.e, directly below `NXentry`) within an application definition or within a NeXus file.

`NXcoordinate_system` can be part of the transformations chain using `NXtransformations`, where it acts as a linear change-of-basis transformation (using a 3x3 matrix with the basis vectors `x`, `y`, and `z` as columns).

Any group that has an optional `depends_on` field or any field that has an optional `depends_on` attribute has a fallback when `depends_on` is not provided. The fallback behavior involves traversing up the hierarchy until the first ancestor that contains one and only one `NXcoordinate_system` group is found. If such an ancestor exists, its coordinate system applies. If none is found or more than one instance of `NXcoordinate_system` is found at the same level, then the current coordinate system is not defined with respect to anything else. As an example, if there is only one `NXcoordinate_system` called “`my_coordinate_system`” defined directly under `NXentry`, each optional `depends_on` field/attribute that is not defined automatically defaults to `depends_on=my_coordinate_system`.

How many groups of type `NXcoordinate_system` should be used in an application definition?

- 0; if there is no instance of `NXcoordinate_system` across the entire tree, you can use `depends_on=". "` to state that this transformation depends on the default NeXus coordinate system (which is the same as the one used by `McStas`).

For the sake of clarity, even in this case it is better to be explicit and consistent for every other coordinate system definition to support users with interpreting the content and logic behind every instance of the tree.

The default NeXus coordinate system (i.e., the `McStas` coordinate system) can be expressed as follows:

```
mcstas@NXcoordinate_system
  x = [1, 0, 0]
  y = [0, 1, 0]
  z = [0, 0, 1]
  @y_direction = "opposite to gravity"
  @z_direction = "direction of the primary beam"
```

Note that this assumes that the direction of the beam is not defined in the NXbeam instance.

- 1; if only one `NXcoordinate_system` is defined, it should be placed as high up in the tree hierarchy (ideally right below an instance of NXentry) of the application definition tree as possible. This coordinate system shall be named such that it is clear how this coordinate system is typically referred to in a community. For the NeXus McStas coordinate system, it is advised to call it `mcstas` for the sake of improved clarity.

If this is the case, it is assumed that this `NXcoordinate_system` overwrites the NeXus default McStas coordinate system, i.e. users can thereby conveniently and explicitly specify the coordinate system that they wish to use.

This case has the advantage that it is explicit and offers no ambiguities. However, in reality typically multiple coordinate systems have to be mastered especially for complex multi-signal modality experiments.

If this case is realized, the best practice is that in every case where this coordinate system should be referred to the respective group has a `depends_on` field, to clearly indicate which specific coordinate systems is referred to.

- 2 and more; as soon as more than one `NXcoordinate_system` is specified somewhere in the tree, different interpretations are possible as to which of these coordinate systems apply or take preference. While these ambiguities should be avoided if possible, the opportunity for multiples instances enables to have coordinate system conventions that are specific for some part of the NeXus tree. This is especially useful for deep groups where multiple scientific methods are combined or cases where having a definition of global conversion tables how to convert between representations in different coordinate systems is not desired or available for now.

To resolve the possible ambiguities which specific coordinate systems in an `NXtransformations` train is referred to, it is even more important to use the `depends_on` field in groups and the `depends_on` attribute in `NXtransformations` to refer to one of the `NXcoordinate_system` instances.

In the case of two or more instances of `NXcoordinate_system` it is advised to specify the relationship between the two coordinate systems by using the `NXtransformations` group within `NXcoordinate_system`.

In any case, users are encouraged to write explicit and clean `depends_on` fields in all groups that encode information for which the interpretation of coordinate systems and transformations is relevant. If these `depends_on` instances are not provided or no instance of `NX_coordinate_system` exists in the upper part of the hierarchy, the application definition is considered underconstrained. Note that this is the case for all files that were created before `NXcoordinate_system` was added.

Symbols:

No symbol table

Groups cited:

`NXtransformations`

Structure:

origin: (optional) `NX_CHAR`

Human-readable field describing where the origin of this CS is. Exemplar values could be *left corner of the lab bench, door handle pinhole through which the electron beam exits the pole piece, barycenter of the triangle, center of mass of the stone*.

type: (optional) `NX_CHAR`

Coordinate system type.

Any of these values or a custom value (if you use a custom value, also set @custom=True):
 undefined | cartesian

x_alias: (optional) *NX_CHAR*

Opportunity to define an alias for the name of the x-axis.

x_direction: (optional) *NX_CHAR*

Human-readable field telling in which direction the x-axis points if that instance of *NXcoordinate_system* has no reference to any parent and as such is the world reference frame.

Exemplar values could be direction of gravity.

x: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

Basis unit vector along the first axis which spans the coordinate system. This axis is frequently referred to as the x-axis in Euclidean space and the i-axis in reciprocal space.

Note that x, y, and z must constitute a basis, i.e., a set of linearly independent vectors that span the vector space.

y_alias: (optional) *NX_CHAR*

Opportunity to define an alias for the name of the y-axis.

y_direction: (optional) *NX_CHAR*

Human-readable field telling in which direction the y-axis points if that instance of *NXcoordinate_system* has no reference to any parent and as such is the mighty world reference frame.

See docstring of **x_direction** for further details.

y: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

Basis unit vector along the second axis which spans the coordinate system. This axis is frequently referred to as the y-axis in Euclidean space and the j-axis in reciprocal space.

z_alias: (optional) *NX_CHAR*

Opportunity to define an alias for the name of the z-axis.

z_direction: (optional) *NX_CHAR*

Human-readable field telling in which direction the z-axis points if that instance of *NXcoordinate_system* has no reference to any parent and as such is the mighty world reference frame.

See docstring of **x_alias** for further details.

z: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

Basis unit vector along the third axis which spans the coordinate system. This axis is frequently referred to as the z-axis in Euclidean space and the k-axis in reciprocal space.

depends_on: (optional) *NX_CHAR*

This specifies the relation to another coordinate system 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 this coordinate system with respect to another coordinate system.

Hypertext Anchors

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

- [*/NXcoordinate_system/depends_on-field*](#)
- [*/NXcoordinate_system/origin-field*](#)
- [*/NXcoordinate_system/TRANSFORMATIONS-group*](#)
- [*/NXcoordinate_system/type-field*](#)
- [*/NXcoordinate_system/x-field*](#)
- [*/NXcoordinate_system/x_alias-field*](#)
- [*/NXcoordinate_system/x_direction-field*](#)
- [*/NXcoordinate_system/y-field*](#)
- [*/NXcoordinate_system/y_alias-field*](#)
- [*/NXcoordinate_system/y_direction-field*](#)
- [*/NXcoordinate_system/z-field*](#)
- [*/NXcoordinate_system/z_alias-field*](#)
- [*/NXcoordinate_system/z_direction-field*](#)

NXDL Source:

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

NXcorrector_cs

Status:

base class, extends [*NXcomponent*](#)

Description:

Base class for a corrector reducing (spherical) aberrations in electron microscopy.

Different technology partners use different conventions and models for quantifying the aberration coefficients.

Correctors in an electron microscope are composed of multiple lenses and multipole stigmators. Their technical details are specific for the technology partner as well as microscope. Most technical details are proprietary knowledge.

If one component corrects for multiple types of aberrations (like it is the case reported here [CEOS](#)) follow this design when using corrector and monochromator in an application definition:

- Use [*NXcorrector_cs*](#) for spherical aberration
- Use [*NXmonochromator*](#) for energy filtering or chromatic aberration
- Use the group corrector_ax in [*NXem*](#) for axial astigmatism aberration

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_img: Number of images taken, at least one.

Groups cited:

[*NXaberration*](#), [*NXaperture*](#), [*NXdeflector*](#), [*NXimage*](#), [*NXlens_em*](#), [*NXprocess*](#)

Structure:**applied:** (optional) *NX_BOOLEAN* <=

Was the corrector used?

tableauID: (optional) *NXprocess*

Specific information about the alignment procedure. This is a process during which the corrector is configured to enable calibrated usage of the instrument.

This *NXprocess* group should also be used when one describes in a computer simulation the specific details about the modelled or assumed aberrations.**description:** (optional) *NX_CHAR*

Discouraged free-text field to add further details about the alignment procedure.

tilt_angle: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_img]) {units=*NX_ANGLE*}

The outer tilt angle of the beam in tableau acquisition.

TODO: The relevant axes which span the tilt_angle need a cleaner description. Suggestions from the community are welcome here for guiding an improvement of this base class.

exposure_time: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_img]) {units=*NX_TIME*}

The exposure time of single tilt images.

magnification: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_img]) {units=*NX_DIMENSIONLESS*}

The factor of enlargement of the apparent size, not the physical size, of an object.

model: (optional) *NX_CHAR*

Convention used for storing measured or estimated aberrations (for each or the final image) via fields c_1, a_1, c_1_0, c_1_2_a, and so on and so forth.

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. Conversion tables are also summarized by Y. Liao.

Any of these values: ceos | nion

imageID: (optional) *NXimage*

Image(s) taken during the alignment procedure

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*
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*

LENS_EM: (optional) *NXlens_em*
APERTURE: (optional) *NXaperture*
DEFLECTOR: (optional) *NXdeflector*

Hypertext Anchors

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

- [*/NXcorrector_cs/APERTURE-group*](#)
- [*/NXcorrector_cs/applied-field*](#)
- [*/NXcorrector_cs/DEFLECTOR-group*](#)
- [*/NXcorrector_cs/LENS_EM-group*](#)
- [*/NXcorrector_cs/tableauID-group*](#)
- [*/NXcorrector_cs/tableauID/a_1-group*](#)
- [*/NXcorrector_cs/tableauID/a_2-group*](#)
- [*/NXcorrector_cs/tableauID/a_3-group*](#)
- [*/NXcorrector_cs/tableauID/a_4-group*](#)
- [*/NXcorrector_cs/tableauID/a_6-group*](#)
- [*/NXcorrector_cs/tableauID/b_2-group*](#)
- [*/NXcorrector_cs/tableauID/b_4-group*](#)
- [*/NXcorrector_cs/tableauID/c_1-group*](#)
- [*/NXcorrector_cs/tableauID/c_1_0-group*](#)
- [*/NXcorrector_cs/tableauID/c_1_2_a-group*](#)
- [*/NXcorrector_cs/tableauID/c_1_2_b-group*](#)
- [*/NXcorrector_cs/tableauID/c_2_1_a-group*](#)
- [*/NXcorrector_cs/tableauID/c_2_1_b-group*](#)
- [*/NXcorrector_cs/tableauID/c_2_3_a-group*](#)
- [*/NXcorrector_cs/tableauID/c_2_3_b-group*](#)
- [*/NXcorrector_cs/tableauID/c_3-group*](#)
- [*/NXcorrector_cs/tableauID/c_3_0-group*](#)
- [*/NXcorrector_cs/tableauID/c_3_2_a-group*](#)
- [*/NXcorrector_cs/tableauID/c_3_2_b-group*](#)
- [*/NXcorrector_cs/tableauID/c_3_4_a-group*](#)
- [*/NXcorrector_cs/tableauID/c_3_4_b-group*](#)
- [*/NXcorrector_cs/tableauID/c_4_1_a-group*](#)
- [*/NXcorrector_cs/tableauID/c_4_1_b-group*](#)
- [*/NXcorrector_cs/tableauID/c_4_3_a-group*](#)
- [*/NXcorrector_cs/tableauID/c_4_3_b-group*](#)
- [*/NXcorrector_cs/tableauID/c_4_5_a-group*](#)
- [*/NXcorrector_cs/tableauID/c_4_5_b-group*](#)
- [*/NXcorrector_cs/tableauID/c_5-group*](#)
- [*/NXcorrector_cs/tableauID/c_5_0-group*](#)

- */NXcorrector_cs/tableauID/c_5_2_a-group*
- */NXcorrector_cs/tableauID/c_5_2_b-group*
- */NXcorrector_cs/tableauID/c_5_4_a-group*
- */NXcorrector_cs/tableauID/c_5_4_b-group*
- */NXcorrector_cs/tableauID/c_5_6_a-group*
- */NXcorrector_cs/tableauID/c_5_6_b-group*
- */NXcorrector_cs/tableauID/d_4-group*
- */NXcorrector_cs/tableauID/description-field*
- */NXcorrector_cs/tableauID/exposure_time-field*
- */NXcorrector_cs/tableauID/imageID-group*
- */NXcorrector_cs/tableauID/magnification-field*
- */NXcorrector_cs/tableauID/model-field*
- */NXcorrector_cs/tableauID/r_5-group*
- */NXcorrector_cs/tableauID/s_3-group*
- */NXcorrector_cs/tableauID/s_5-group*
- */NXcorrector_cs/tableauID/tilt_angle-field*

NXDL Source:

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

NXcrystal

Status:

base class, extends *NXcomponent*

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*

Structure:

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* <=

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

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/GEOOMETRY-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/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*

NXDL Source:

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

NXcs_computer**Status:**

base class, extends *NXObject*

Description:

Base class for reporting the description of a computer

Symbols:

No symbol table

Groups cited:

NXcs_memory, *NXcs_process*, *NXcs_storage*

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*

A globally unique persistent identifier of the computer, i.e. the Universally Unique Identifier (UUID) of the computing node.

CS_PROCESS: (optional) *NXcs_process*

CS_MEMORY: (optional) *NXcs_memory*

CS_STORAGE: (optional) *NXcs_storage*

Hypertext Anchors

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

- */NXcs_computer/CS_MEMORY-group*
- */NXcs_computer/CS_PROCESS-group*
- */NXcs_computer/CS_STORAGE-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/base_classes/NXcs_computer.nxdl.xml

NXcs_memory

Status:

base class, extends *NXcomponent*

Description:

Base class for reporting the description of the memory system of a computer.

Symbols:

No symbol table

Groups cited:

NXcircuit

Structure:

CIRCUIT: (optional) *NXcircuit*

type: (optional) *NX_CHAR*

Qualifier for the type of random access memory.

Any of these values or a custom value (if you use a custom value, also set @custom=True): ddr4 | ddr5

max_physical_capacity: (optional) *NX_POSINT* {units=*NX_ANY*}

Total amount of data which the medium can hold.

Hypertext Anchors

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

- */NXcs_memory/CIRCUIT-group*
- */NXcs_memory/CIRCUIT/max_physical_capacity-field*
- */NXcs_memory/CIRCUIT/type-field*

NXDL Source:

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

NXcs_process

Status:

base class, extends *NXcomponent*

Description:

Base class for reporting the description of processing units of a computer.

Examples are e.g. (classical) processing units (CPUs), coprocessor, graphic cards, accelerator processing units or a system of these.

Symbols:

No symbol table

Groups cited:

NXcircuit

Structure:

CIRCUIT: (optional) *NXcircuit*

Typical examples for the granularization of processing units are:

- A desktop computer with a single CPU; describe using one instance of NXcircuit.
- A dual-socket server; describe using two instances of NXcircuit.
- A server with two dual-socket server nodes; describe with four instances of NXcircuit surplus a field that defines their level in the hierarchy.

type: (optional) *NX_CHAR*

General type of the processing unit

Any of these values or a custom value (if you use a custom value, also set @custom=True): cpu | gpu | fpga

Hypertext Anchors

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

- [/NXcs_process/CIRCUIT-group](#)
- [/NXcs_process/CIRCUIT/type-field](#)

NXDL Source:

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

NXcs_profiling

Status:

base class, extends [NXobject](#)

Description:

Computer science description for performance and 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 and these are used usually by multiple users.

At the most basic level users may wish to document how long e.g. a data analysis with a scientific software (app) took. A frequent idea is here to answer practical questions like how critical is the effect 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? For this more qualitative performance monitoring, mainly the order of magnitude is relevant, as well as how this was achieved using parallelization (i.e. reporting the number of CPU and GPU resources used, the number of processes and threads configured, and providing basic details about the 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, i.e. by software and hardware developers as well as for tracking bugs. 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 for now. 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 demands a more precise record of the wall-clock time.

number_of_processes: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Qualifier which specifies with how many nominal processes the app was invoked. The main idea behind this field e.g. for apps which use e.g. MPI (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_UINT* {units=*NX_UNITLESS*}

Qualifier how many nominal threads were used 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_UINT* {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/base_classes/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 for a more precise record of the wall-clock time.

Elapsed time may contain time portions where resources were idling.

number_of_processes: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of processes used (max) during the execution of this event.

number_of_threads: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of threads used (max) during the execution of this event.

number_of_gpus: (optional) *NX_UINT* {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/base_classes/NXcs_profiling_event.nxdl.xml

NXcs_storage

Status:

base class, extends *NXcomponent*

Description:

Base class for reporting the description of the I/O of a computer.

Symbols:

No symbol table

Groups cited:

NXcircuit

Structure:

CIRCUIT: (optional) *NXcircuit*

type: (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_POSINT* {units=*NX_ANY*}

Total amount of data which the medium can hold.

Hypertext Anchors

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

- [/NXcs_storage/CIRCUIT-group](#)
- [/NXcs_storage/CIRCUIT/max_physical_capacity-field](#)
- [/NXcs_storage/CIRCUIT/type-field](#)

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/base_classes/NXcs_storage.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 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:

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`

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. NX-data groups contain plottable data (also referred to as *signals* or *dependent variables*) and their associated axis coordinates (also 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>

Usage**Curve**

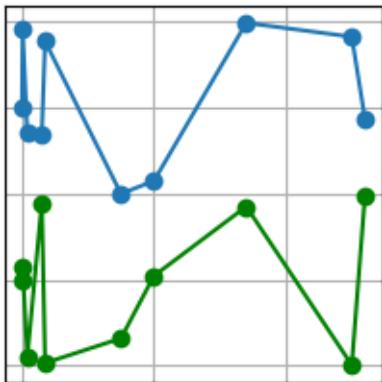
One-dimensional curves as a function of the same axis.

The x-axis defines a regular 1D grid. As illustrated in this example, the grid is not necessarily equally spaced.

```
@NX_class = "NXroot"
@default = "scan1"
scan1:NXentry
  @NX_class = "NXentry"
  @default = "data"
  data:NXdata
    @NX_class = "NXdata"
    @auxiliary_signals = ["y2"]
    @axes = ["x"]
    @signal = "y1"
    x:NX_INT64[10]
    y1:NX_FLOAT64[10]
    y2:NX_FLOAT64[10]
```

Explanation:

1. `@axes` has one value which corresponds to the signal rank of one.
2. `y1` is the default signal to be plotted versus `x`.
3. `y2` is the only alternative signal to be plotted versus `x`.



```
# Data
import numpy as np

x = [0, 1, 8, 30, 35, 150, 200, 340, 500, 520]
y1 = 4 + np.sin(2 * np.array(x))
y2 = 2 + np.sin(3 * np.array(x))

# Plot
import matplotlib.pyplot as plt # noqa E402

plt.style.use("_mpl-gallery")

fig, ax = plt.subplots()

ax.plot(x, y1, "o-", label="y1")
ax.plot(x, y2, "go-", label="y2")

plt.show()
```

Total running time of the script: (0 minutes 0.043 seconds)

2D Scatter

Scatter data in two dimensions.

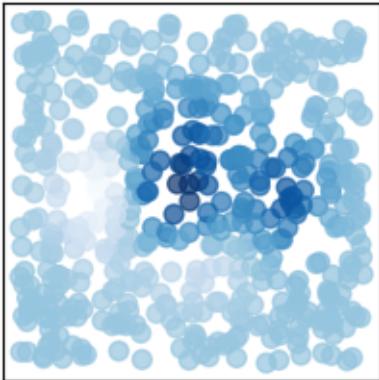
The x-axis and y-axis define the coordinates of the scattered points.

```
@NX_class = "NXroot"
@default = "scan1"
scan1:
    @NX_class = "NXentry"
    @default = "data"
    data:
        @NX_class = "NXdata"
        @x_indices = [0]
        @y_indices = [0]
        @signal = "z"
        x: NX_FLOAT64[500]
        y: NX_FLOAT64[500]
        z: NX_FLOAT64[500]
```

Explanation:

1. `@axes` is omitted since default axes are not applicable.
2. `z` is the default signal to be plotted versus `x` and `y`.
3. Each data point in `z` has a coordinate in `x` and `y`.

Note that the NXdata structure does **not** describe how the data needs to be plotted. In this example we create a 2D scatter plot in the XY-plane. But the data could also be plotted as a function of `x` or `y` in a *Curve* plot. When defining `@axes="x"` the curve plot would be the most likely intention of the data publisher but it is up to the reader to decide how to plot the data.



```
# Data
import numpy as np

rstate = np.random.RandomState(42)
x = rstate.uniform(-3, 3, 500)
y = rstate.uniform(-3, 3, 500)
z = (1 - x / 2 + x**5 + y**3) * np.exp(-(x**2) - y**2)

# Plot
import matplotlib.pyplot as plt # noqa E402

plt.style.use("_mpl-gallery-nogrid")

fig, ax = plt.subplots()
ax.scatter(x, y, c=z, s=50, alpha=0.7)

plt.show()
```

Total running time of the script: (0 minutes 0.054 seconds)

Image

Image on a regular two-dimensional grid.

The x-axis and y-axis define a regular 2D grid. The grid is equally spaced in this example but this is not necessarily the case.

```
@NX_class = "NXroot"
@default = "scan1"
scan1:
    @NX_class = "NXentry"
    @default = "data"
```

(continues on next page)

(continued from previous page)

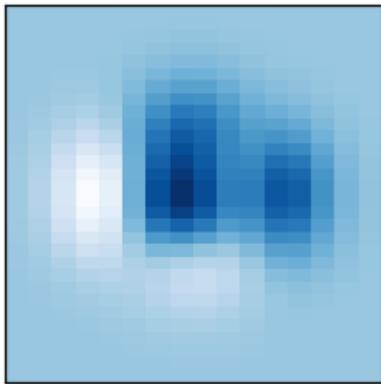
```

data:
@NX_class = "NXdata"
@axes = ["y", "x"]
@signal = "z"
x: NX_FLOAT64[16]
y: NX_FLOAT64[30]
z: NX_FLOAT64[30,16]

```

Explanation:

1. @axes has two values which corresponds to the signal rank of two.
2. z is the default signal to be plotted versus x and y.
3. z has 30 rows and 16 columns.
4. y spans the first dimension of z and x the second.



```

# Data
import numpy as np

x = np.linspace(-3, 3, 16)
y = np.linspace(-3, 3, 30)

xx, yy = np.meshgrid(x, y)
z = (1 - xx / 2 + xx**5 + yy**3) * np.exp(-(xx**2) - yy**2)

# Plot
import matplotlib.pyplot as plt # noqa E402

plt.style.use("_mpl-gallery-nogrid")

fig, ax = plt.subplots()

ax.imshow(z, extent=[x[0], x[-1], y[0], y[-1]], origin="lower")

plt.show()

```

Total running time of the script: (0 minutes 0.031 seconds)

1D Histogram

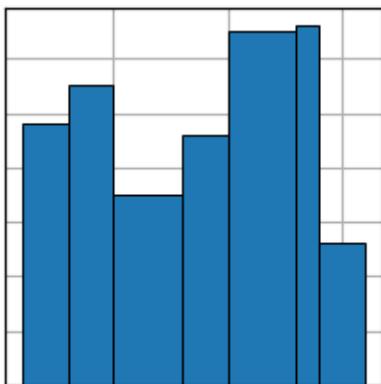
Histogram on a regular one-dimensional grid.

The x-axis defines the bin edges. As illustrated in this example, the bin widths are not necessarily identical.

```
@NX_class = "NXroot"
@default = "scan1"
scan1:
  @NX_class = "NXentry"
  @default = "data"
  data:
    @NX_class = "NXdata"
    @axes = ["x"]
    @signal = "y"
    x: NX_INT64[8]
    y: NX_FLOAT64[7]
```

Explanation:

1. `@axes` has one value which corresponds to the signal rank of one.
2. `y` is the default signal to be plotted versus `x`.
3. `x` has one more value than `y` since it contains the bin edges.



```
# Data
x = [0.5, 1.5, 2.5, 4, 5, 6.5, 7, 8]
y = [4.8, 5.5, 3.5, 4.6, 6.5, 6.6, 2.6]

# Plot
import numpy as np # noqa E402
import matplotlib.pyplot as plt # noqa E402

plt.style.use("_mpl-gallery")

fig, ax = plt.subplots()

centers = 0.5 * (np.array(x[:-1]) + np.array(x[1:]))
widths = np.diff(x)
ax.bar(centers, y, width=widths, edgecolor="k", linewidth=0.7)

plt.show()
```

Total running time of the script: (0 minutes 0.036 seconds)

2D Histogram

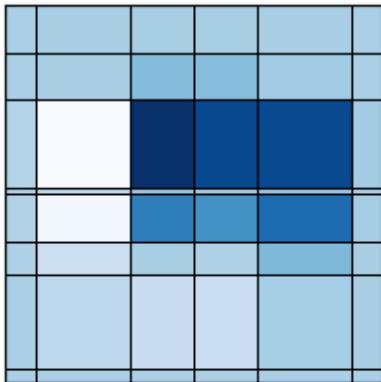
Histogram on a regular two-dimensional grid.

The x-axis and y-axis define the bin edges. As illustrated in this example, the bin widths are not necessarily identical.

```
@NX_class = "NXroot"
@default = "scan1"
scan1:
  @NX_class = "NXentry"
  @default = "data"
  data:
    @NX_class = "NXdata"
    @axes = ["y", "x"]
    @signal = "z"
    x: NX_FLOAT64[7]
    y: NX_FLOAT64[9]
    z: NX_FLOAT64[8,6]
```

Explanation:

1. `@axes` has two values which corresponds to the signal rank of two.
2. `z` is the default signal to be plotted versus `x` and `y`.
3. `z` has 6 rows and 8 columns.
4. `y` has one more value than the first dimension of `z` since it contains the bin edges.
5. `x` has one more value than the second dimension of `z` since it contains the bin edges.



```
# Data
import numpy as np

x = [-3.0, -2.5, -1.0, 0.0, 1.0, 2.5, 3.0]
y = [-3.0, -2.8, -1.3, -0.75, 0.0, 0.1, 1.5, 2.25, 3.0]

xx = np.linspace(-3, 3, 200)
yy = np.linspace(-3, 3, 200)
xx, yy = np.meshgrid(xx, yy)
zz = (1 - xx / 2 + xx**5 + yy**3) * np.exp(-(xx**2) - yy**2)
```

(continues on next page)

(continued from previous page)

```

z, _, _ = np.histogram2d(yy.flatten(), xx.flatten(), bins=[y, x], weights=zz.
    ↪flatten())

# Plot
import matplotlib.pyplot as plt # noqa E402

plt.style.use("_mpl-gallery-nogrid")

fig, ax = plt.subplots()

mesh = ax.pcolormesh(x, y, z, edgecolor="k", linewidth=0.7, shading="flat")

plt.show()

```

Total running time of the script: (0 minutes 0.035 seconds)

2D Mixed-Histogram

Data on a regular two-dimensional grid which is the combination of an image and a histogram.

The x-axis defines the bin edges along the second dimension. The y-axis defines the grid coordinates along the first dimension.

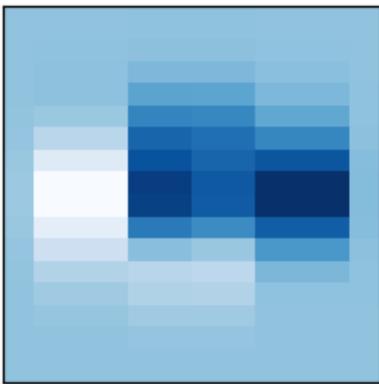
```

@NX_class = "NXroot"
@default = "scan1"
scan1:
  @NX_class = "NXentry"
  @default = "data"
  data:
    @NX_class = "NXdata"
    @axes = ["y", "x"]
    @signal = "z"
    x: NX_FLOAT64[7]
    y: NX_FLOAT64[16]
    z: NX_FLOAT64[16,6]

```

Explanation:

1. `@axes` has two values which corresponds to the signal rank of two.
2. `z` is the default signal to be plotted versus `x` and `y`.
3. `z` has 16 rows and 6 columns.
4. `x` has one more value than the second dimension of `z` since it contains the bin edges.



```
# Data
import numpy as np

x = [-3.0, -2.5, -1.0, 0.0, 1.0, 2.5, 3.0]
y = np.linspace(-3, 3, 16)

nx = len(x) - 1
ny = len(y)
z = np.zeros((ny, nx))
xx = np.linspace(-3, 3, 200)
for i in range(ny):
    zi = (1 - xx / 2 + xx**5 + y[i] ** 3) * np.exp(-(xx**2) - y[i] ** 2)
    z[i, :] = np.histogram(xx, bins=x, weights=zi)

# Plot
import matplotlib.pyplot as plt # noqa E402

plt.style.use('_mpl-gallery-nogrid')

fig, ax = plt.subplots()

y_new = np.empty_like(y, shape=(len(y) + 1,))
y_new[0] = 2 * y[0] - y[1]
y_new[1:-1] = (y[:-1] + y[1:]) / 2
y_new[-1] = 2 * y[-1] - y[-2]

mesh = ax.pcolormesh(x, y_new, z, linewidth=0.7)

plt.show()
```

Total running time of the script: (0 minutes 0.033 seconds)

2D Continuous scan

Data on a regular two-dimensional grid which is the combination of an image and a histogram.

The x-axis defines the bin edges along the second dimension. The y-axis defines the grid coordinates along the first dimension.

This is a typical 2D scanning example where a sample is scanned through a focussed X-ray beam in two dimensions. The x motor is scanned continuously and both motors have an encoder which records the actual motor position as opposed to the requested or *set* position.

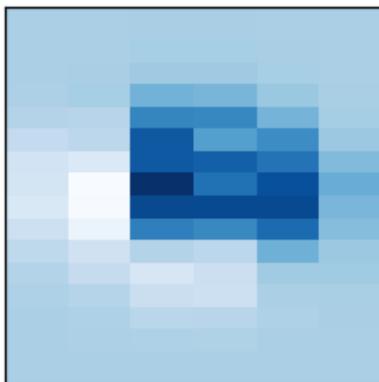
```

@NX_class = "NXroot"
@default = "scan1"
scan1:
  @NX_class = "NXentry"
  @default = "data"
  data:
    @NX_class = "NXdata"
    @axes = ["y_set", "x_set"]
    @x_encoder_indices = [0, 1]
    @y_encoder_indices = 0
    @signal = "z"
    z: NX_FLOAT64[16, 6]
    x_encoder: NX_FLOAT64[16, 7]
    y_encoder: NX_FLOAT64[16]
    x_set: NX_FLOAT64[6]
    y_set: NX_FLOAT64[16]

```

Explanation:

1. `@axes` has two values which corresponds to the signal rank of two.
2. `x_set` and `y_set` are the default axes which can be used by readers that cannot handle multi-dimensional coordinates.
3. `x_set_indices` and `y_set_indices` are omitted because they would be equal to the position of "`x_set`" and "`y_set`" in `@axes`.
4. The first dimension is spanned by three axes `y_set`, `y_encoder` and `x_encoder`. The axes `y_set` and `y_encoder` have as many values as there are data points along the first dimension. This is because the y motor moves one step after each scan of the x motor.
5. The second dimension is spanned by two axes `x_set` and `x_encoder`. Since the x motor is scanned continuously, the encoder records the edge of every bin on which an data is recorded yielding 7 values instead of 6 along the second dimension.
6. The `x_encoder` spans the first dimensions because the actual x edges are slightly different for every x motion at each y position. Since the y motor does not move while scanning x, there is no need for `y_encoder` to span the second dimension because the value along this dimension remains constant.



```

# Data
import numpy as np

x_set = np.linspace(-3, 3, 7)

```

(continues on next page)

(continued from previous page)

```

y_set = np.linspace(-3, 3, 16)

rstate = np.random.RandomState(42)
noise_x = 0.1 * (x_set[1] - x_set[0])
noise_y = 0.1 * (y_set[1] - y_set[0])
x_encoder = x_set[np.newaxis, :] + rstate.normal(0, noise_x, (len(y_set), ↴
    len(x_set)))
y_encoder = y_set + rstate.normal(0, noise_y, len(y_set))

nx = len(x_set) - 1
ny = len(y_set)
z = np.zeros((ny, nx))
xx = np.linspace(-3, 3, 200)
for i in range(ny):
    zi = (1 - xx / 2 + xx**5 + y_encoder[i]**3) * np.exp(-(xx**2) - y_←
        encoder[i]**2)
    z[i, :] = np.histogram(xx, bins=x_encoder[i, :], weights=zi)

# Plot
import matplotlib.pyplot as plt # noqa E402

plt.style.use("_mpl-gallery-nogrid")

fig, ax = plt.subplots()

y_set_new = np.empty_like(y_set, shape=(len(y_set) + 1,))
y_set_new[0] = 2 * y_set[0] - y_set[1]
y_set_new[1:-1] = (y_set[:-1] + y_set[1:]) / 2
y_set_new[-1] = 2 * y_set[-1] - y_set[-2]

mesh = ax.pcolormesh(x_set, y_set_new, z, linewidth=0.7)
plt.show()

```

Total running time of the script: (0 minutes 0.033 seconds)

Example of a simple curve plot

```

data:NXdata
@signal = "data"
@axes = ["x"]
data: float[100]
x: float[100]

```

More complex cases are supported

- histogram data: x has one more value than data.
- alternative axes: instead of a single x axis you can have several axes, one of which being the default.
- signals with more than one dimension: data could be 2D with axes x and y along each dimension.
- axes with more than one dimension: data could be 2D with axes x and y also being 2D, providing

a unique (x, y) coordinate for each data point.

Signals:

Defined by

- *DATA* fields
- the *signal* attribute
- the *auxiliary_signals* attribute

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:

Defined by

- *AXISNAME* fields
- the *axes* attribute
- *AXISNAME_indices* attributes

The fields and attributes are defined as follows

1. The *AXISNAME* fields contain the axis coordinates associated with the signal values.
2. The *axes* attribute provides the names of the *AXISNAME* fields to be used as the *default axis* for each dimension of the *DATA* fields.
3. The *AXISNAME_indices* attributes describe the *DATA* dimensions spanned by the corresponding *AXISNAME* fields.

The fields and attributes have the following constraints

1. The length of the *axes* attribute must be equal to the rank of the *DATA* fields. When a particular dimension has no default axis, the string ":" is used in that position.
2. The number of values in *AXISNAME_indices* must be equal to the rank of the corresponding *AXISNAME* field.
3. When *AXISNAME_indices* is missing for a given *AXISNAME* field, the positions of the *AXISNAME* field name in the *axes* attribute are used.

4. When `AXISNAME_indices` is the same as the indices of “AXISNAME” in the `axes` attribute, there is no need to provide `AXISNAME_indices`.
5. The indices of “AXISNAME” in the `axes` attribute must be a subset of `AXISNAME_indices`.
6. The shape of an `AXISNAME` field must correspond to the shape of the `DATA` dimensions it spans. This means that for each dimension `i` in `[0, AXISNAME.ndim]` spanned by axis field `AXISNAME`, the number of axis values `AXISNAME.shape[i]` along dimension `i` must be equal to the number of data points `DATA.shape[AXISNAME_indices[i]]` along dimension `i` or one more than the number of data points `DATA.shape[AXISNAME_indices[i]]+1` in case the `AXISNAME` field contains histogram bin edges along dimension `i`.

Highlight consequences of these constraints

1. An `AXISNAME` field can have more than one dimension and can therefore span more than one `DATA` dimension. Conversely, one `DATA` dimension can be spanned by more than one `AXISNAME` field. The default axis name (if any) of each dimension can be found in the `axes` attribute.
2. A list of all available axes is not provided directly. All strings in the `axes` attribute (excluding the “.” string) are axis field names. In addition the prefix of an attribute ending with the string “_indices” is also an axis field name.

The following example covers all axes features supported (see 2D Continuous scan)

```
data:NXdata
@signal = "data"
@axes = ["x_set", "y_set", "."] # default axes for all three dimensions
@x_encoder_indices = [0, 1]           # or [1]
@y_encoder_indices = 1
data: float[10,7,1024]
x_encoder: float[11,7]               # coordinates along the first and second
→dimensions
y_encoder: float[7]                 # coordinates along the second dimension
x_set: float[10]                     # default coordinates along the first
→dimension
y_set: float[7]                     # default coordinates along the second
→dimension
```

Uncertainties:

Defined by

- `FIELDNAME_errors` fields

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"]
```

```

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]
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** dimensions are spanned by the corresponding axis. The first dimension index is 0 (zero).

The number of indices must be equal to the rank of the **AXISNAME** field.

When the **AXISNAME_indices** attribute is missing for a given **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 to be used as the default axis along every **DATA** dimension. As a result the length must be equal to the rank of the **DATA** fields. The string “.” can be used for dimensions without a default axis.

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 shape of an *AXISNAME* field must correspond to the shape of the *DATA* dimensions it spans. This means that for each *i* in $[0, \text{AXISNAME}.ndim)$ the number of data points *DATA*.
shape[*AXISNAME*_indices[i]] must be equal to the number of coordinates *AXISNAME*.
shape[i] or the number of bin edges *AXISNAME*.shape[i]+1 in case of histogram data.

As the upper case *AXISNAME* indicates, the names of the *AXISNAME* fields can be chosen *freely*.

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. This can be a [DATA](#) field (signal or auxiliary signal) or a [AXISNAME](#) field (axis).

The dimensions of the FIELDNAME_errors field must match the dimensions of the corresponding 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.

FIELDNAME_scaling_factor: (optional) [NX_NUMBER](#)

An optional scaling factor to apply to the values in any field named FIELDNAME in this NXdata group. This can be a [DATA](#) field (signal or auxiliary signal) or a [AXISNAME](#) field (axis).

The elements stored in NXdata datasets are often stored as integers for efficiency reasons and need further correction or conversion, generating floats. For example, raw values could be stored from a device that need to be converted to values that represent the physical values. The two fields FIELDNAME_scaling_factor and FIELDNAME_offset allow linear corrections using the following convention:

```
corrected values = (FIELDNAME + offset) * scaling_factor
```

This formula will derive the values to use in downstream applications, when necessary.

When omitted, the scaling factor is assumed to be 1.

FIELDNAME_offset: (optional) [NX_NUMBER](#)

An optional offset to apply to the values in FIELDNAME (usually the signal).

When omitted, the offset is assumed to be 0.

See [FIELDNAME_scaling_factor](#) for more information.

scaling_factor: (optional) [NX_FLOAT](#)

DEPRECATED: Use FIELDNAME_scaling_factor instead

The scaling_factor and FIELDNAME_scaling_factor fields have similar semantics. However, scaling_factor is ambiguous in the case of multiple signals. Therefore scaling_factor is deprecated. Use FIELDNAME_scaling_factor instead, even when only a single signal is present.

offset: (optional) [NX_FLOAT](#)

DEPRECATED: Use FIELDNAME_offset instead

The offset and FIELDNAME_offset fields have similar semantics. However, offset is ambiguous in the case of multiple signals. Therefore offset is deprecated. Use FIELDNAME_offset instead, even when only a single signal is present.

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 compatibility.

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 compatibility.

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 compatibility.

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/FIELDNAME_offset-field*
- */NXdata/FIELDNAME_scaling_factor-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

NXdeflector

Status:

base class, extends *NXcomponent*

Description:

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

Symbols:

No symbol table

Groups cited:

none

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 `NXfabrication` and `identifierNAME`.

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.

Hypertext Anchors

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

- [*/NXdeflector/current-field*](#)
- [*/NXdeflector/name-field*](#)
- [*/NXdeflector/type-field*](#)
- [*/NXdeflector/voltage-field*](#)

NXDL Source:

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

NXdetector

Status:

base class, extends *NXcomponent*

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

k: number of detector pixels in the third (if necessary, fastest) 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*

Structure:

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 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 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
- pulse counting

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 measurements and this way of recording the different measurements preserves the chronological order with is important for correct processing.

This is used for example in tomography ([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 $\text{countrate_correction_lookup_table}[c]$.

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* <=

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: 3, Dimensions: [i, j, k])
{units=*NX_DIMENSIONLESS*}

efficiency of the detector

wavelength: (optional) *NX_FLOAT* (Rank: 3, Dimensions: [i, j, k])
{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.

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/raw_time_of_flight@frequency-attribute`
- `/NXdetector/real_time-field`
- `/NXdetector/saturation_value-field`
- `/NXdetector/sensor_material-field`
- `/NXdetector/sensor_thickness-field`
- `/NXdetector/sequence_number-field`
- `/NXdetector/serial_number-field`
- `/NXdetector/slot-field`
- `/NXdetector/slot@local_name-attribute`
- `/NXdetector/solid_angle-field`
- `/NXdetector/start_time-field`
- `/NXdetector/start_time@start-attribute`
- `/NXdetector/stop_time-field`
- `/NXdetector/stop_time@start-attribute`
- `/NXdetector/threshold_energy-field`
- `/NXdetector/time_of_flight-field`

- */NXdetector/time_of_flight@axis-attribute*
- */NXdetector/time_of_flight@long_name-attribute*
- */NXdetector/time_of_flight@primary-attribute*
- */NXdetector/trigger_dead_time-field*
- */NXdetector/trigger_delay_time-field*
- */NXdetector/trigger_delay_time_set-field*
- */NXdetector/trigger_internal_delay_time-field*
- */NXdetector/type-field*
- */NXdetector/underload_value-field*
- */NXdetector/virtual_pixel_interpolation_applied-field*
- */NXdetector/x_pixel_offset-field*
- */NXdetector/x_pixel_offset@axis-attribute*
- */NXdetector/x_pixel_offset@long_name-attribute*
- */NXdetector/x_pixel_offset@primary-attribute*
- */NXdetector/x_pixel_size-field*
- */NXdetector/y_pixel_offset-field*
- */NXdetector/y_pixel_offset@axis-attribute*
- */NXdetector/y_pixel_offset@long_name-attribute*
- */NXdetector/y_pixel_offset@primary-attribute*
- */NXdetector/y_pixel_size-field*
- */NXdetector/z_pixel_offset-field*
- */NXdetector/z_pixel_offset@axis-attribute*
- */NXdetector/z_pixel_offset@long_name-attribute*
- */NXdetector/z_pixel_offset@primary-attribute*

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:

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*

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:**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*

NXDL Source:

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

NXdisk_chopper

Status:

base class, extends *NXcomponent*

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*

Structure:

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, anti-clockwise 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.

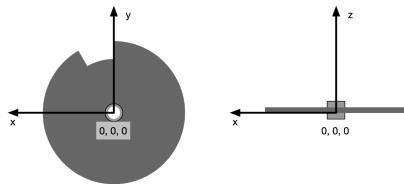
wavelength_range: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [2]) {units=*NX_WAVELENGTH*}

Low and high values of wavelength range transmitted

depends_on: (optional) *NX_CHAR* <=

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

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/type-field*](#)

- */NXdisk_chopper/wavelength_range-field*

NXDL Source:

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

NXdistortion**Status:**

base class, extends *NXprocess*

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:

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/original_centre-field*](#)
- [*/NXdistortion/original_points-field*](#)
- [*/NXdistortion/rdeform_field-field*](#)
- [*/NXdistortion/symmetry-field*](#)

NXDL Source:

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

NXebeam_column

Status:

base class, extends [*NXcomponent*](#)

Description:

Base class for a set of components providing a controllable electron beam.

The idea behind defining NXebeam_column as an own base class vs. adding these concepts in NXinstrument_em is that the electron beam generating component might be worthwhile to use also in other types of experiments.

Symbols:

No symbol table

Groups cited:

[*NXactuator*](#), [*NXaperture*](#), [*NXbeam*](#), [*NXcomponent*](#), [*NXcorrector_cs*](#), [*NXdeflector*](#), [*NXfabrication*](#), [*NXlens_em*](#), [*NXmonochromator*](#), [*NXscanbox_em*](#), [*NXsensor*](#), [*NXsource*](#)

Structure:

operation_mode: (optional) [*NX_CHAR*](#)

Tech-partner, microscope-, and control-software-specific name of the specific operation mode how the ebeam_column and its components are controlled to achieve specific illumination conditions.

In many cases the users of an instrument do not or can not be expected to know all intricate spatiotemporal dynamics of their hardware. Instead, they rely on assumptions that the instrument, its control software, and components work as expected to focus on their research questions.

For these cases, having a place for documenting the operation_mode is useful in as much as at least some constraints on how the illumination conditions were documented.

electron_source: (optional) [*NXsource*](#)

A physical part of an electron or ion microscope from which the particles that form the beam are emitted.

The hardware for an electron source in an electron microscope may contain several components which affect the beam path.

This concept is related to term [Source](#) of the EMglossary standard.

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

The potential difference between anode and cathode.

This concept is related to term [Acceleration Voltage](#) of the EMglossary standard.

extraction_voltage: (optional) *NX_NUMBER* {units=*NX_VOLTAGE*}

Voltage which is used to create an electric field that draws particles from the source.

This concept is related to term [Extraction Voltage](#) of the EMglossary standard.

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

Electrical current which is released from the source.

This concept is related to term [Emission Current](#) of the EMglossary standard.

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

Electrical current which flows through the source.

This concept is related to term [Filament Current](#) of the EMglossary standard.

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.

emitter_material: (optional) *NX_CHAR*

Material of which the emitter is build, e.g. the filament material.

lifetime: (optional) *NX_NUMBER* {units=*NX_TIME*}

How long has the source been in operation.

LENS_EM: (optional) *NXlens_em*

APERTURE: (optional) *NXaperture*

DEFLECTOR: (optional) *NXdeflector*

blankerID: (optional) *NXdeflector*

A component for blanking the beam or generating pulsed electron beams. See e.g . [I. G. C. Weppelman et al.](#) or [Y. Liao](#) for details.

MONOCHROMATOR: (optional) *NXmonochromator*

Device to improve energy resolution or chromatic aberration.

Examples are Wien, \$textalpa\$-, or \$Omega\$- energy filter or [cc](#) corrector like

type: (optional) *NX_CHAR*

Qualitative type of the component.

Any of these values:

- [wien](#)

- alfa
- omega
- castaing_henry
- gatan_imaging
- sector_analyzer

applied: (optional) *NX_BOOLEAN* <=

Was the corrector used?

dispersion: (optional) *NX_NUMBER* {units=*NX_ANY*}

Energy dispersion in e.g. $\mu\text{m}/\text{eV}$.

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

Corresponding voltage for that energy dispersion.

FABRICATION: (optional) *NX_fabrication* <=

CORRECTOR_CS: (optional) *NX_corrector_cs*

corrector_ax: (optional) *NXcomponent*

Component that reshapes an ellipse-shaped electron beam into a circular one.

- L. Reimer 1998, Springer, 1998
- M. Tanaka et al., Electron Microscopy Glossary, 2024

Stigmator is an exact synonym.

value_x: (optional) *NX_NUMBER* {units=*NX_ANY*}

Descriptor for the correction strength along the first direction when exact technical details are unknown or not directly controllable as the control software of the microscope does not enable or was not configured to display these values for users.

value_y: (optional) *NX_NUMBER* {units=*NX_ANY*}

Descriptor for the correction strength along the second direction when exact technical details are unknown or not directly controllable as the control software of the microscope does not enable or was not configured to display these values for users.

biprismID: (optional) *NXcomponent*

Electron biprism as it is used e.g. for electron holography.

phaseplateID: (optional) *NXcomponent*

Device that causes a change in the phase of an electron wave.

- M. Malac et al.
- R. R. Schröder et al.

type: (optional) *NX_CHAR*

Qualitative type

Any of these values or a custom value (if you use a custom value, also set @custom=True): `thin_film` | `electrostatic`

SENSOR: (optional) *NXsensor*

ACTUATOR: (optional) *NXactuator*

BEAM: (optional) *NXbeam*

Individual characterization results for the position, shape, and characteristics of the electron beam at a given location.

NXtransformations should be used to specify the location or the position at which details about the beam were probed.

This concept is related to term **Electron Beam** of the EMglossary standard.

COMPONENT: (optional) *NXcomponent*

scan_controller: (optional) *NXscanbox_em*

Hypertext Anchors

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

- */NXbeam_column/ACTUATOR-group*
- */NXbeam_column/APERTURE-group*
- */NXbeam_column/BEAM-group*
- */NXbeam_column/biprismID-group*
- */NXbeam_column/blankerID-group*
- */NXbeam_column/COMPONENT-group*
- */NXbeam_column/corrector_ax-group*
- */NXbeam_column/corrector_ax/value_x-field*
- */NXbeam_column/corrector_ax/value_y-field*
- */NXbeam_column/CORRECTOR_CS-group*
- */NXbeam_column/DEFLECTOR-group*
- */NXbeam_column/electron_source-group*
- */NXbeam_column/electron_source/emission_current-field*
- */NXbeam_column/electron_source/emitter_material-field*
- */NXbeam_column/electron_source/emitter_type-field*
- */NXbeam_column/electron_source/extraction_voltage-field*
- */NXbeam_column/electron_source/filament_current-field*
- */NXbeam_column/electron_source/lifetime-field*
- */NXbeam_column/electron_source/probe-field*
- */NXbeam_column/electron_source/voltage-field*
- */NXbeam_column/LENS_EM-group*
- */NXbeam_column/MONOCHROMATOR-group*
- */NXbeam_column/MONOCHROMATOR/applied-field*
- */NXbeam_column/MONOCHROMATOR/dispersion-field*

- */NXbeam_column/MONOCHROMATOR/FABRICATION-group*
- */NXbeam_column/MONOCHROMATOR/type-field*
- */NXbeam_column/MONOCHROMATOR/voltage-field*
- */NXbeam_column/operation_mode-field*
- */NXbeam_column/phaseplateID-group*
- */NXbeam_column/phaseplateID/type-field*
- */NXbeam_column/scan_controller-group*
- */NXbeam_column/SENSOR-group*

NXDL Source:

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

NXelectron_detector

Status:

base class, extends *NXdetector*

Description:

A subclass of NXdetector for detectors that detect electrons.

Symbols:

No symbol table

Groups cited:

none

Structure:

amplifier_type: (optional) *NX_CHAR*

Type of electron amplifier, MCP, channeltron, etc.

detector_type: (optional) *NX_CHAR*

Description of the electron detector type, DLD, Phosphor+CCD, CMOS.

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

Voltage applied to the electron detector.

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

Voltage applied to the amplifier.

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

The low voltage of the amplifier might not be the ground.

Hypertext Anchors

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

- */NXelectron_detector/amplifier_bias-field*
- */NXelectron_detector/amplifier_type-field*
- */NXelectron_detector/amplifier_voltage-field*
- */NXelectron_detector/detector_type-field*

- */NXelectron_detector/detector_voltage_field*

NXDL Source:

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

NXem_ebsd**Status:**

base class, extends *NXprocess*

Description:

Base class method-specific for Electron Backscatter Diffraction (EBSD).

The general procedure of an EBSD experiment is as follows: Users load the specimen, collect first a coarse image of the surface. Next, they set an approximate value for the calibrated working distance and tilt the stage into diffraction conditions.

Users then typically configure the microscope for collecting quality data. The EBSD detector is pushed in (if retractable). Subsequently, they fine tune the illumination and aberration corrector settings and select one or multiple ROIs for the microscope to machine off automatically. They configure on-the-fly indexing parameter and then typically start the measurement queue. From this point onwards typically the microscope runs automatically.

Diffraction pattern get collected until the queue finishes or gets interrupted by either errors or arrival at the end of the users' allocated timeslot at the instrument.

Kikuchi pattern (EBSP) are usually indexed on-the-fly. These patterns are the raw data. Once indexed, these patterns are often not stored.

Results are stored in files, which afterwards are typically copied automatically or manually for archival purposes to certain storage locations for further consumption. The result of such an EBSD measurement/experiment is a set of usually proprietary or open files from technology partners.

This *NXem_ebsd* base class is a proposal how to represent method-specific data, metadata, and connections between these for the research field of electron microscopy exemplified here for electron backscatter diffraction (EBSD). The base class solves two key documentation issues within the EBSD community:

Firstly, an instance of NXem_ebsd (such as a NeXus/HDF5 file that is formatted according to NXem_ebsd) stores the connection between the microscope session and the key datasets which are considered typically results of the afore-mentioned steps involved in an EBSD experiment.

Different groups in NXem_ebsd make connections to data artifacts which were collected when working with electron microscopes via the NXem 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 acquisition and eventual indexing of Kikuchi patterns, associated overview images, like secondary electron or backscattered electron images of the region-of-interest probed, and many more (meta)data.

Secondly, NXem_ebsd connects and stores the conventions and reference frames which were used and which are the key to a correct mathematical interpretation of every experiment or simulation using EBSD.

Otherwise, results would be ripped out of their context like it is the current situation with many traditional studies where EBSD data were indexed on-the-fly and shared with the community only via sharing the strongly processed files with results in some formatting but without communicating all conventions used or just relying on the assumptions that colleagues likely know these conventions even though multiple definitions are possible.

NXem_ebsd 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) experiments where a combination of repetitive removal of material from the surface layer to measure otherwise the same region-of-interest at different depth increments. Material removal can be achieved with mechanical, electron, or ion polishing, using manual steps or automated equipment like a robot system S. Tsai et al..

Three-dimensional experiments require to follow a sequence of specimen, surface preparation, and data collection steps. By virtue of design, these methods are destructive either because of the necessary material removal or surface degradation 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 via 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 contextualized and connected.

Eventual tomography methods also use such a workflow because first diffraction images are collected (e.g. with X-ray) and then these images are indexed to process a 3D orientation mapping. Therefore, the here proposed base class can be a blueprint also for future classes to embrace our colleagues from X-ray-based techniques be it 3DXRD or HEDM.

This concept is related to term [Electron Backscatter Diffraction](#) of the EMglossary standard.

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 plot orientation mapping.

n_y: Number of pixel along slow changing dimension for a rediscretized i.e. standardized default plot orientation mapping.

n_x: Number of pixel along fast changing dimension for a rediscretized i.e. standardized default plot orientation mapping.

n_solutions: Number of phase solutions

n_hkl: Number of reflectors (Miller crystallographic plane triplets).

Groups cited:

[NXcollection](#), [NXcoordinate_system](#), [NXdata](#), [NXnote](#), [NXphase](#), [NXprocess](#), [NXrotations](#)

Structure:

gnomonic_reference_frame: (optional) [NXcoordinate_system](#)

Details about the gnomonic (projection) reference frame.

It is assumed that the configuration is inspected by looking towards the sample surface. If a detector is involved, it is assumed that the configuration is inspected from a position that is located behind this detector.

If any of these assumptions are not met, the user is required to explicitly state this.

Reference <https://doi.org/10.1016/j.matchar.2016.04.008> suggests to label the base vectors of this coordinate system as X_g , Y_g , Z_g .

origin: (optional) [NX_CHAR](#) <=

Origin of the gnomonic_reference_frame.

Reference <https://doi.org/10.1016/j.matchar.2016.04.008> suggests to assume that this is coordinate $X_g = 0$, $Y_g = 0$, $Z_g = 0$.

Obligatory value: `in_the_pattern_center`

x_direction: (optional) `NX_CHAR <=`

Direction of the positively pointing x-axis base vector of the gnomonic_reference_frame.

Any of these values:

- north
- east
- south
- west
- in
- out

y_direction: (optional) `NX_CHAR <=`

Direction of the positively pointing y-axis base vector of the gnomonic_reference_frame.

Any of these values:

- north
- east
- south
- west
- in
- out

z_direction: (optional) `NX_CHAR <=`

Direction of the positively pointing z-axis base vector of the gnomonic_reference_frame.

Any of these values:

- north
- east
- south
- west
- in
- out

pattern_center: (optional) `NXprocess`

Details about the definition of the pattern center as a special point in the gnomonic_reference_frame.

Typically the gnomonic space is embedded in the detector space. Specifically, the XgYg plane is defined such that it is 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 one effectively normalizes in fractions of the width of the detector.

The issue with terms like width and height, though, is that these become degenerated if the detector region-of-interest is square-shaped. This is why instead of referring to width and height it is better to state explicitly which direction is considered positive when measuring distances.

For the concepts used to specify the boundary_convention it is assumed that the region-of-interest is defined by a rectangle, referring to the direction of outer-unit normals to the respective edges of this rectangle.

x_boundary_convention: (optional) [NX_CHAR](#)

From which border of the EBSP (in the detector reference frame) is the pattern center's x-position (PCx) measured.

Any of these values: `top | right | bottom | left`

x_normalization_direction: (optional) [NX_CHAR](#)

In which direction are positive values for the x-axis coordinate value measured from the specified boundary.

Any of these values: `north | east | south | west`

y_boundary_convention: (optional) [NX_CHAR](#)

From which border of the EBSP (in the detector reference frame) is the pattern center's y-position (PCy) measured.

Any of these values: `top | right | bottom | left`

y_normalization_direction: (optional) [NX_CHAR](#)

In which direction are positive values for the y-axis coordinate value measured from the specified boundary.

Any of these values: `north | east | south | west`

measurement: (optional) [NXprocess](#)

This group documents relevant details about the conditions and the tools for measuring diffraction patterns with an electron microscope.

The most frequently collected EBSD data are captured for rectangular regions-of-interest using a discretization into square or hexagon tiles.

time: (optional) [NX_NUMBER](#) {units=[NX_TIME](#)}

Physical time since the beginning of a timestamp that is required to be the same for all experiments in the set. The purpose of this marker is to identify how all experiments in the set need 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, 30 minutes, 6 hours, or 24 hours of annealing.

@epoch_start: (optional) [NX_CHAR](#)

Timestamp relative to which time was counted to aid converting between time and timestamp.

depends_on: (optional) [NX_CHAR](#)

Path to an instance of [NXdata](#) where the measured patterns are stored.

source: (optional) [NXnote <=](#)

Reference (e.g. path and filename) to an existent data artifact which stores either the measured patterns or input (already processed EBSD data).

simulation: (optional) [NXprocess](#)

This group documents relevant details about the conditions and the tools used for simulating diffraction patterns with some physical model.

This group should be used if (e.g. instead of a measurement) the patterns were simulated (possibly awaiting indexing).

In many practical cases where patterns are analyzed on-the-fly and dictionary indexing strategies used, so-called master pattern(s) are used to compare measured or simulated patterns with the master patterns.

depends_on: (optional) [NX_CHAR](#)

Path to an instance of [NXimage](#) where the simulated patterns are stored.

source: (optional) [NXnote <=](#)

Reference (e.g. path and filename) to an existent digital resource which stores either the patterns or input (already processed EBSD data) that are about to become processed further as described by this NXem_ebsd instance.

calibration: (optional) [NXprocess](#)

The EBSD system, including components like the electron gun, pole-piece, stage tilt, EBSD detector, and the gnomonic projection have to be calibrated to achieve reliable, precise, and accurate scientific results.

Specifically, the gnomonic projection has to be calibrated. Typically, standard specimens made from silicon or quartz crystals in specific orientations are used for this purpose.

Considering that a system used is already calibrated well-enough is much more frequently the case in practice than that users perform the calibration themselves (with above-mentioned standard specimens).

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 has been calibrated already. Consequently, users pick from an existent library of phase candidates, i.e. [NXunit_cell](#) instances. Examples are reflector models as stored in CRY files (HKL/Channel 5/Flamenco).

In the second case, users calibrate the system during the session using standards (silicon, quartz, or other common specimens). There is usually one person in each lab responsible for doing such calibrations. Often this person or technician is 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: Taking TSL OIM/EDAX as an example, the conventions how orientations are stored is affected by how the reference frames are configured and how this setup in the GUI.

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' obligation to remember these settings or write these down in a lab notebook. Otherwise, these metadata get lost. All these issues are a motivation and problem which [NXem_ebsd](#) solves in that all conventions can be specified explicitly.

depends_on: (optional) [NX_CHAR](#)

Path to an instance of [NXem](#) where calibration data are stored.

source: (optional) [NXnote <=](#)

Reference to a digital resource where the calibration is stored.

indexing: (optional) [NXprocess](#)

Indexing is a data processing step performed either after or while (aka on-the-fly) the beam scans the specimen. The resulting method is also known as orientation imaging microscopy (OIM).

Different algorithms can be used to index EBSP. Common to them is the computational step where simulated or theoretically assumed patterns are compared with the measured ones. These latter patterns are referred to via the measurement or simulation groups of this base class respectively.

Quality descriptors are defined based on which an indexing algorithm yields a quantitative measure of how similar measured and reference patterns are, and thus if no, one, or multiple so-called solutions were found.

Assumed or simulated patterns are simulated using kinematical or dynamical theory of electron diffraction delivering master patterns.

The Hough transform, one of the most frequently used traditional method for indexing EBSP is essentially a discretized Radon transform (for details see [M. van Ginkel et al.](#)). Recently, dictionary-based and artificial intelligence-based methods find more widespread usage for indexing.

method: (optional) [NX_CHAR](#)

Principal algorithm used for indexing.

Any of these values or a custom value (if you use a custom value, also set @custom=True): `hough_transform` | `dictionary` | `radon_transform`

status: (optional) [NX_UINT](#) (Rank: 1, Dimensions: [n_sc]) {units=[NX_UNITLESS](#)}

Which return value did the indexing algorithm yield for each scan point.

- 0 - Not analyzed
- 1 - Too high angular deviation
- 2 - No solution
- 100 - Success
- 255 - Unexpected errors

phases_per_scan_point: (optional) [NX_INT](#) (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 [NXunit_cell](#).

In another example users may have skipped some scan points (not indexed them at all) or used differing numbers of phases for indexing different scan points.

The cumulated of this array decodes how identifier_phase and matching_phase 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), identifier_phase has as many entries as scan points and matching_phase has also as many entries as scan points.

identifier_phase: (optional) [NX_INT](#) (Rank: 1, Dimensions: [n_solutions]) {units=[NX_UNITLESS](#)}

The array phases_per_scan_point details how the identifier_phase and the matching_phase arrays have to be interpreted.

For the example with a single phase identifier_phase 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 required) that a pattern matches eventually (not equally well) sufficiently significant with multiple patterns. This can especially happen in cases of pseudosymmetry and more frequently with an improperly calibrated system or false or inaccurate phase models. Having such field is especially relevant for recent dictionary- or artificial intelligence-based indexing methods to communicate the results in a model-agnostic way in combination with matching_phase.

Depending on the phases_per_scan_point value, identifier_phase and matching_phase arrays represent a collection of concatenated tuples. These 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 phases_per_scan_point.

matching_phase: (optional) *NX_INT* (Rank: 1, Dimensions: [n_solutions])
{units=*NX_UNITLESS*}

One-dimensional array, pattern by pattern labelling the solutions found. The array phases_per_scan_point has to be specified because it details how the identifier_phase and the matching_phase arrays are interpreted. See documentation of identifier_phase for further details.

matching_phase_descriptor: (optional) *NX_CHAR*

Phase_matching is a descriptor for how well the solution matches or not. Examples can be confidence_index, mean_angular_deviation, or other.

Any of these values:

- confidence_index
- mean_angular_deviation
- other

scan_point_positions: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_sc, 2])
{units=*NX_LENGTH*}

Calibrated center positions of each scan point in the sample surface reference system.

indexing_rate: (optional) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

Fraction of successfully indexed patterns with a phase not the null-phase vs the number_of_scan_points.

number_of_scan_points: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of scan points in the original mapping.

source: (optional) *NXnote <=*

This group enables to establish a logical connection between previous processing steps or on-the-fly-performed indexing of the EBSD map. Typically these processing steps are performed with commercial software. Therefore, in many cases a results file from this indexing is often all that is communicated and saved. These are typically files in a format specific to the instrument and its configuration.

Typical file formats are CPR/CRC, ANG, OSC, HDF5, H5EBSD, EDAXH5.

background_correction: (optional) *NXprocess*

Details about the background correction applied to each Kikuchi pattern.

binning: (optional) *NXprocess*

Binning i.e. downsampling to each pattern.

parameter: (optional) *NXcollection*

Specific parameter relevant only for certain algorithms used.

phaseID: (optional) *NXphase*

Details for each phase used as a model with which the patterns were indexed. Instances of *NXunit_cell* in this group must have the group name prefixed with phase. The identifier in the name is an integer. Start counting from 1 because the value 0 is reserved for the special phase that is the null-model, the null phase also known as notIndexed.

dspacing: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_hkl])
{units=*NX_LENGTH*}

Spacing between the crystallographic planes that are defined via *miller*.

relative_intensity: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_hkl])
{units=*NX_DIMENSIONLESS*}

Relative intensity for the computed diffraction intensity (signal) for the plane.

number_of_scan_points: (optional) *NX_UINT* {units=*NX_UNITLESS*}

In case the *NXunit_cell* base class is used with analyzed orientation maps this field stores how many scan points of the map were identified as matching best with this phase.

number_of_planes: (optional) *NX_UINT* {units=*NX_UNITLESS*}

How many reflectors for crystallographic planes are distinguished.

miller: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_hkl, 6])
{units=*NX_UNITLESS*}

Miller indices (*hkl*)[*uvw*] of the planes.

The first triplet specifies (*hkl*). The second triplet specifies [*uvw*]. Miller indices refer to the Cartesian right-handed coordinate system of the unit cell.

rotation: (optional) *NXrotations*

roi: (optional) *NXdata*

An overview of the entire ROI.

descriptor: (optional) *NX_CHAR*

Descriptor representing the image contrast.

Any of these values:

- `band_contrast`
- `confidence_index`
- `mean-angular_deviation`

title: (optional) *NX_CHAR* <=

Title of the default plot.

data: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_y, n_x])
 {units=*NX_UNITLESS*} <=

Descriptor values displaying the ROI.

@long_name: (optional) *NX_CHAR* <=

Descriptor values

axis_y: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_y])
 {units=*NX_LENGTH*}

Calibrated coordinate along the y-axis.

@long_name: (optional) *NX_CHAR*

Label for the y axis

axis_x: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_x])
 {units=*NX_LENGTH*}

Calibrated coordinate along the x-axis.

@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.

- */NXem_ebsd/calibration-group*
- */NXem_ebsd/calibration/depends_on-field*
- */NXem_ebsd/calibration/source-group*
- */NXem_ebsd/gnomonic_reference_frame-group*
- */NXem_ebsd/gnomonic_reference_frame/origin-field*
- */NXem_ebsd/gnomonic_reference_frame/x_direction-field*
- */NXem_ebsd/gnomonic_reference_frame/y_direction-field*
- */NXem_ebsd/gnomonic_reference_frame/z_direction-field*
- */NXem_ebsd/indexing-group*
- */NXem_ebsd/indexing/background_correction-group*
- */NXem_ebsd/indexing/binning-group*
- */NXem_ebsd/indexing/identifier_phase-field*
- */NXem_ebsd/indexing/indexing_rate-field*
- */NXem_ebsd/indexing/matching_phase-field*
- */NXem_ebsd/indexing/matching_phase_descriptor-field*
- */NXem_ebsd/indexing/method-field*
- */NXem_ebsd/indexing/number_of_scan_points-field*
- */NXem_ebsd/indexing/parameter-group*
- */NXem_ebsd/indexing/phaselD-group*

- */NXem_ebsd/indexing/phaseID/dspacing-field*
- */NXem_ebsd/indexing/phaseID/miller-field*
- */NXem_ebsd/indexing/phaseID/number_of_planes-field*
- */NXem_ebsd/indexing/phaseID/number_of_scan_points-field*
- */NXem_ebsd/indexing/phaseID/relative_intensity-field*
- */NXem_ebsd/indexing/phases_per_scan_point-field*
- */NXem_ebsd/indexing/roi-group*
- */NXem_ebsd/indexing/roi/axis_x-field*
- */NXem_ebsd/indexing/roi/axis_x@long_name-attribute*
- */NXem_ebsd/indexing/roi/axis_y-field*
- */NXem_ebsd/indexing/roi/axis_y@long_name-attribute*
- */NXem_ebsd/indexing/roi/data-field*
- */NXem_ebsd/indexing/roi/data@long_name-attribute*
- */NXem_ebsd/indexing/roi(descriptor-field*
- */NXem_ebsd/indexing/roi/title-field*
- */NXem_ebsd/indexing/rotation-group*
- */NXem_ebsd/indexing/scan_point_positions-field*
- */NXem_ebsd/indexing/source-group*
- */NXem_ebsd/indexing/status-field*
- */NXem_ebsd/measurement-group*
- */NXem_ebsd/measurement/depends_on-field*
- */NXem_ebsd/measurement/source-group*
- */NXem_ebsd/measurement/time-field*
- */NXem_ebsd/measurement/time@epoch_start-attribute*
- */NXem_ebsd/pattern_center-group*
- */NXem_ebsd/pattern_center/x_boundary_convention-field*
- */NXem_ebsd/pattern_center/x_normalization_direction-field*
- */NXem_ebsd/pattern_center/y_boundary_convention-field*
- */NXem_ebsd/pattern_center/y_normalization_direction-field*
- */NXem_ebsd/simulation-group*
- */NXem_ebsd/simulation/depends_on-field*
- */NXem_ebsd/simulation/source-group*

NXDL Source:

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

NXem_eds**Status:**

base class, extends [NXprocess](#)

Description:

Base class method-specific for energy-dispersive X-ray spectroscopy (EDS/EDXS).

IUPAC instead of Siegbahn notation should be used.

X-ray spectroscopy is a surface-sensitive technique. Therefore, three-dimensional elemental characterization requires typically a sequence of characterization and preparation of the surface to expose new surface layer that can be characterized in the next acquisition. In effect, the resulting three-dimensional elemental information mappings are truly the result of a correlation and post-processing of several measurements which is the field of correlative tomographic usage of electron microscopy.

Symbols:

n_photon_energy: Number of X-ray photon energy (bins)

n_elements: Number of identified elements

n_peaks: Number of peaks detected

n_iupac_line_names: Number of IUPAC line names

Groups cited:

[NXatom](#), [NXdata](#), [NXimage](#), [NXpeak](#), [NXprocess](#), [NXprogram](#)

Structure:

indexing: (optional) [NXprocess](#)

Details about computational steps how peaks were indexed as elements.

atom_types: (optional) [NX_CHAR](#)

Comma-separated list of symbols for elements from the periodic table that have been confirmed present by the here reported EDS analysis.

This field can be used when creating instances of [NXpeak](#) is not desired. However, a collection of instances of NXpeak with individual NXatom can be used to add isotopic information and other relevant context.

PROGRAM: (optional) [NXprogram](#)

The program with which the indexing was performed.

summary: (optional) [NXdata](#)

Accumulated intensity over all pixels of the region-of-interest.

intensity: (optional) [NX_NUMBER](#) (Rank: 1, Dimensions: [n_photon_energy])
{units=[NX_UNITLESS](#)}

Accumulated counts

@long_name: (optional) [NX_CHAR](#)

Counts

axis_energy: (optional) [NX_NUMBER](#) (Rank: 1, Dimensions: [n_photon_energy])
{units=[NX_ENERGY](#)}

Energy axis

@long_name: (optional) *NX_CHAR*

Energy

PEAK: (optional) *NXpeak*

Details about individual indexed peaks.

ATOM: (optional) *NXatom*

energy_range: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [2])
{units=*NX_ENERGY*}

Associated lower [e_{min}, e_{max}] bounds of the energy which is assumed associated with this peak.

energy: (optional) *NX_NUMBER* {units=*NX_ENERGY*}

Theoretical energy of the line according to IUPAC.

iupac_line_name: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_iupac_line_names])

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 grouped with the same peak.

ELEMENT_SPECIFIC_MAP: (optional) *NXimage*

Individual element-specific EDS/EDX/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.

These element-specific EDS maps are instances of *NXimage* that should be named by the element from the atom_types field.

When signal contributions from several peaks were decomposed users should ideally use a respective number of NXpeak instances to give further context about the individual signal contributions are summarized and shown together, e.g. the combined signal under the curve of carbon and oxygen.

In this case specify the processing details use peak and weight.

description: (optional) *NX_CHAR*

Discouraged free-text field to add additional information.

iupac_line_candidates: (optional) *NX_CHAR*

Comma-separated list of chemical_symbol-IUPAC X-ray (emission) line name that documents which elements and their specific lines are theoretically located within the energy_range of the spectrum from which the EDS (element) map was computed.

energy_range: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [2])
{units=*NX_ENERGY*}

Associated [e_{min}, e_{max}] bounds of the energy range for which spectrum counts were accumulated.

PROCESS: (optional) *NXprocess* <=

peak: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_peaks])

A list of `NXpeak` instance names whose X-ray quanta were accumulated for each pixel to obtain an element-specific EDS map.

weight: (optional) `NX_NUMBER` {units=`NX_UNITLESS`}

A list of weights by how much the intensity of each peak contributes to the intensity of the EDS map.

Hypertext Anchors

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

- `/NXem_eds/indexing-group`
- `/NXem_eds/indexing/atom_types-field`
- `/NXem_eds/indexing/ELEMENT_SPECIFIC_MAP-group`
- `/NXem_eds/indexing/ELEMENT_SPECIFIC_MAP/description-field`
- `/NXem_eds/indexing/ELEMENT_SPECIFIC_MAP/energy_range-field`
- `/NXem_eds/indexing/ELEMENT_SPECIFIC_MAP/iupac_line_candidates-field`
- `/NXem_eds/indexing/ELEMENT_SPECIFIC_MAP/PROCESS-group`
- `/NXem_eds/indexing/ELEMENT_SPECIFIC_MAP/PROCESS/peak-field`
- `/NXem_eds/indexing/ELEMENT_SPECIFIC_MAP/PROCESS/weight-field`
- `/NXem_eds/indexing/PEAK-group`
- `/NXem_eds/indexing/PEAK/ATOM-group`
- `/NXem_eds/indexing/PEAK/ATOM/energy-field`
- `/NXem_eds/indexing/PEAK/ATOM/energy_range-field`
- `/NXem_eds/indexing/PEAK/ATOM/iupac_line_name-field`
- `/NXem_eds/indexing/PROGRAM-group`
- `/NXem_eds/indexing/summary-group`
- `/NXem_eds/indexing/summary/axis_energy-field`
- `/NXem_eds/indexing/summary/axis_energy@long_name-attribute`
- `/NXem_eds/indexing/summary/intensity-field`
- `/NXem_eds/indexing/summary/intensity@long_name-attribute`

NXDL Source:

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

NXem_eels

Status:

base class, extends `NXprocess`

Description:

Base class method-specific for Electron Energy Loss Spectroscopy (EELS).

Symbols:

No symbol table

Groups cited:

NXpeak, NXprocess, NXprogram, NXspectrum

Structure:

zlp_correction: (optional) *NXprocess*

Details about computational steps how the zero-loss peak was threaded.

PROGRAM: (optional) *NXprogram*

The program with which the zero-loss peak correction was performed.

indexing: (optional) *NXprocess*

Details about computational steps how peaks were indexed as elements.

PROGRAM: (optional) *NXprogram*

The program with which the indexing was performed.

PEAK: (optional) *NXpeak*

Name and location of each peak in the spectrum considered to be of relevance.

SPECTRUM: (optional) *NXspectrum*

NXspectrum specialized for EELS.

Hypertext Anchors

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

- */NXem_eels/indexing-group*
- */NXem_eels/indexing/PEAK-group*
- */NXem_eels/indexing/PROGRAM-group*
- */NXem_eels/indexing/SPECTRUM-group*
- */NXem_eels/zlp_correction-group*
- */NXem_eels/zlp_correction/PROGRAM-group*

NXDL Source:

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

NXem_img

Status:

base class, extends *NXprocess*

Description:

Base class for method-specific generic imaging with electron microscopes.

In the majority of cases simple d-dimensional regular scan patterns are used to probe regions-of-interest (ROIs). Examples can be single point aka spot measurements, line profiles, or (rectangular) surface map-pings. 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:

No symbol table

Groups cited:*NXimage***Structure:**IMAGE: (optional) *NXimage***imaging_mode:** (optional) *NX_CHAR*

Which imaging mode was used?

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- secondary_electron
- backscattered_electron
- annular_dark_field
- cathodoluminescence

half_angle_interval: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [2])
{units=*NX_ANGLE*}

Annulus inner (first value) and outer (second value) half angle.

Hypertext Anchors

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

- */NXem_img/IMAGE-group*
- */NXem_img/IMAGE/half_angle_interval-field*
- */NXem_img/IMAGE/imaging_mode-field*

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

NXem_measurement

Status:base class, extends *NXObject***Description:**

Base class for documenting a measurement with an electron microscope.

Symbols:

No symbol table

Groups cited:*NXevent_data_em*, *NXinstrument_em***Structure:****instrument:** (optional) *NXinstrument_em***eventID:** (optional) *NXevent_data_em*

Hypertext Anchors

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

- [*/NXem_measurement/eventID-group*](#)
- [*/NXem_measurement/instrument-group*](#)

NXDL Source:

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

NXem_simulation

Status:

base class, extends [*NXObject*](#)

Description:

Base class for documenting a simulation of electron beam-matter interaction.

Symbols:

No symbol table

Groups cited:

[*NXdata*](#), [*NXparameters*](#), [*NXprocess*](#), [*NXprogram*](#)

Structure:

PROGRAM: (optional) [*NXprogram*](#)

PARAMETERS: (optional) [*NXparameters*](#)

PROCESS: (optional) [*NXprocess*](#)

DATA: (optional) [*NXdata*](#)

Hypertext Anchors

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

- [*/NXem_simulation/DATA-group*](#)
- [*/NXem_simulation/PARAMETERS-group*](#)
- [*/NXem_simulation/PROCESS-group*](#)
- [*/NXem_simulation/PROGRAM-group*](#)

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/base_classes/NXem_simulation.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 Plot-table 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 meeting¹, 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 permissible 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*

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*

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

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:

NXactuator, *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

value: (optional) *NX_FLOAT* {units=*NX_ANY*}

This is to be used if there is no actuator/sensor that controls/measures the environment parameters, but the user would still like to give a value for it. An example would be a room temperature experiment where the temperature is not actively measured, but rather estimated.

Note that this method for recording the environment parameters is not advised, but using NXsensor and NXactuator is strongly recommended 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.

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

ACTUATOR: (optional) [NXactuator](#)

Any actuator used to control the environment. This can be linked to an actuator defined in an NXinstrument instance.

SENSOR: (optional) [NXsensor](#)

Any sensor used to monitor the environment. This can be linked to a sensor defined in an NXinstrument instance.

Hypertext Anchors

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

- [/NXenvironment/ACTUATOR-group](#)
- [/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](#)
- [/NXenvironment/value-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:

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](#)

NXDL Source:

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

NXevent_data_em

Status:

base class, extends [NXobject](#)

Description:

Base class to store state and (meta)data of events for electron microscopy.

Event-related (meta)data, typically measured datasets like images and spectra. To avoid repetitively storing static instrument-related metadata, the dynamic (meta)data that typically changes for each image and spectrum is split from the static (meta)data.

Which temporal granularity is adequate to log events depends on the situation and research question. Using a model which enables a collection of events offers the most flexible way to cater for both experiments with controlled electron beams in a real microscope or the simulation of such experiments or individual aspects of such experiments.

Electron microscopes are dynamic. Scientists often report that microscopes *perform differently* across sessions. That *they* perform differently from one day or another. In some cases, root causes for performance differences are unclear. Users of the instrument may consider such conditions impractical, or *too poor*, and thus abort their session. Alternatively, users may try to bring the microscope into a state where conditions are considered better or of whatever high enough quality for starting or continuing the measurement.

In all these use cases it is useful to have a mechanism whereby time-dependent data of the instrument state can be stored and documented in a representation that facilitates interoperability. This is the idea behind this base class.

NXevent_data_em represents an instance to describe and serialize flexibly whatever is considered a time interval during which the instrument is considered stable enough for allowing any working on tasks with it. Examples of such tasks are the collecting of data (images and spectra) or the calibrating the instrument or individual of its components. Users may wish to take only a single scan or image and complete their session thereafter. Alternatively, users are working for much longer time at the instrument, perform recalibrations in between and take several scans (of different ROIs on the specimen), or they explore the state of the microscope for service or maintenance tasks.

NXevent_data_em serves the harmonization and documentation of these cases:

- Firstly, via a header section whose purpose is to contextualize and identify the event instance in time.
- Secondly, via a data and metadata section where individual data collections can be stored in a standardized representation.

We are aware of the fact that given the variety how an electron microscope is used, there is a need for a flexible and adaptive documentation system. At the same time we are also convinced though that just because one has different requirements for some specific aspect under the umbrella of settings to an electron microscope, this does not necessarily warrant that one has to cook up an own data schema.

Instead, the electron microscopy community should work towards reusing schema components as frequently as possible. This will enable that there is at all not only a value of harmonizing electron microscopy research content but also there is a technical possibility to build services around such harmonized data.

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 a stack of spectra. Having to deal with instabilities is a common theme in electron microscopy practice. Numerical protocols can be used during data post-processing to correct for some of the instabilities. A few exemplar references provide an overview on the subject:

- 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 (digital twin), 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 and ions, or detector noise models. This also warrants to document the time-dependent details of individual components of the microscope via the here implemented class *NXevent_data_em*.

Symbols:

No symbol table

Groups cited:

NXimage, *NXinstrument_em*, *NXspectrum*, *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 users wish 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.

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

Identifier of a specific state and setting of the microscope.

identifier_sample: (optional) *NX_CHAR* {units=*NX_UNITLESS*}

The name of the sample to resolve ambiguities.

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 for which there is at the moment no other place.

In the long run such free-text field description should be avoided as it is difficult to machine-interpret. Instead, an enumeration should be used.

USER: (optional) [NXuser](#)

INSTRUMENT_EM: (optional) [NXinstrument_em](#)

IMAGE: (optional) [NXimage](#)

SPECTRUM: (optional) [NXspectrum](#)

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/identifier_event-field](#)
- [/NXevent_data_em/identifier_sample-field](#)
- [/NXevent_data_em/IMAGE-group](#)
- [/NXevent_data_em/INSTRUMENT_EM-group](#)
- [/NXevent_data_em/SPECTRUM-group](#)
- [/NXevent_data_em/start_time-field](#)
- [/NXevent_data_em/type-field](#)
- [/NXevent_data_em/USER-group](#)

NXDL Source:

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

NXfabrication**Status:**

base class, extends *NXObject*

Description:

Details about a component as it is 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.

@version: (optional) *NX_CHAR*

If it is possible that different versions exist, the value in this field should be made specific enough to resolve the version.

serial_number: (optional) *NX_CHAR*

Serial number of the component.

construction_date: (optional) *NX_DATE_TIME*

Datetime of component's initial construction. This refers to the date of first measurement after new construction or to the relocation date, if it describes a multicomponent/custom-build setup. Just the year is often sufficient, but if a full date/time is used, it is recommended to add an explicit time zone.

capability: (optional) *NX_CHAR*

Free-text list 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/construction_date-field*
- */NXfabrication/model-field*
- */NXfabrication/model@version-attribute*
- */NXfabrication/serial_number-field*
- */NXfabrication/vendor-field*

NXDL Source:

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

NXfermi_chopper**Status:**

base class, extends *NXcomponent*

Description:

A Fermi chopper, possibly with curved slits.

Symbols:

No symbol table

Groups cited:

NXgeometry, *NXoff_geometry*

Structure:

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* <=

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

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/transmitting_material-field*
- */NXfermi_chopper/type-field*
- */NXfermi_chopper/wavelength-field*
- */NXfermi_chopper/width-field*

NXDL Source:

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

NXfilter

Status:

base class, extends *NXcomponent*

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](#)

Structure:

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* <=

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

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/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*

NXDL Source:

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

NXflipper**Status:**

base class, extends *NXcomponent*

Description:

A spin flipper.

Symbols:

No symbol table

Groups cited:

none

Structure:

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* <=

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/type-field*

NXDL Source:

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

NXfresnel_zone_plate**Status:**

base class, extends *NXcomponent*

Description:

A fresnel zone plate

Symbols:

No symbol table

Groups cited:

none

Structure:

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* <=

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/zone_height-field*
- */NXfresnel_zone_plate/zone_material-field*
- */NXfresnel_zone_plate/zone_support_material-field*

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:

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*

NXDL Source:

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

NXgrating

Status:

base class, extends *NXcomponent*

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:

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* <=

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*

NXDL Source:

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

NXguide

Status:

base class, extends *NXcomponent*

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

Structure:

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*}

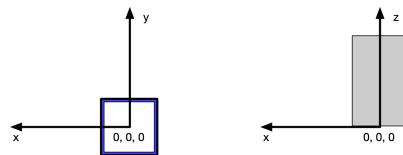
TODO: documentation needed

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* <=

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

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*

NXDL Source:

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

NXhistory**Status:**

base class, extends *NXObject*

Description:

A set of activities that occurred to a physical entity prior/during experiment.

Ideally, a full report of the previous operations (or links to a chain of operations). Alternatively, notes allow for additional descriptors in any format.

Symbols:

No symbol table

Groups cited:

NXactivity, *NXnote*

Structure:

identifierNAME: (optional) *NX_CHAR*

An ID or reference to the location or a unique (globally persistent) identifier of e.g. another file which gives as many as possible details of the history event.

ACTIVITY: (optional) *NXactivity*

Any activity that was performed on the physical entity prior or during the experiment.

NOTE: (optional) *NXnote*

A descriptor to keep track of the treatment of the physical entity before or during the experiment (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. This should only be used in case that there is no rigorous description using the base classes above. This group can also be used to pull in any activities that are not well described by an existing base class definition.

Any number of instances of NXnote are allowed for describing extra details of this activity.

Hypertext Anchors

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

- [*/NXhistory/ACTIVITY-group*](#)
- [*/NXhistory/identifierNAME-field*](#)
- [*/NXhistory/NOTE-group*](#)

NXDL Source:

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

NXbeam_column

Status:

base class, extends [*NXcomponent*](#)

Description:

Base class for a set of components equipping an instrument with FIB capabilities.

Focused-ion-beam (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. The beam is controlled such that it is intense, focused, and equipped with sufficient ion having sufficient momentum to remove material in a controlled manner.

The fact that an electron microscope with FIB capabilities achieves these functionalities via a second component (aka the ion gun) that has its own relevant control circuits, focusing lenses, and other components, warrants the definition of 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](#)
- [N. Yao](#)

Symbols:

No symbol table

Groups cited:

[*NXactuator*](#), [*NXaperture*](#), [*NXbeam*](#), [*NXcomponent*](#), [*NXdeflector*](#), [*NXion*](#), [*NXlens_em*](#), [*NXmonochromator*](#), [*NXscanbox_em*](#), [*NXsensor*](#), [*NXsource*](#)

Structure:

operation_mode: (optional) [*NX_CHAR*](#)

Tech-partner, microscope-, and control-software-specific name of the specific operation mode how the ibeam_column and its components are controlled to achieve specific illumination conditions.

In many cases the users of an instrument do not or can not be expected to know all intricate spatiotemporal dynamics of their hardware. Instead, they rely on assumptions that the instrument, its control software, and components work as expected to focus on their research questions.

For these cases, having a place for documenting the operation_mode is useful in as much as at least some constraints on how the illumination conditions were documented.

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.

flux: (optional) *NX_NUMBER* {units=*NX_ANY*}

Average/nominal flux

brightness: (optional) *NX_NUMBER* {units=*NX_ANY*}

Average/nominal brightness

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

Charge current

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

Ion acceleration voltage upon source exit and entering the vacuum flight path.

ion_energy_profile: (optional) *NX_NUMBER* {units=*NX_ENERGY*}

To be defined more specifically. Community suggestions are welcome.

probe: (optional) *NXion*

Which ionized elements or molecular ions form the beam. Examples are gallium, helium, neon, argon, krypton, or xenon, O₂+

LENS_EM: (optional) *NXlens_em*

APERTURE: (optional) *NXaperture*

DEFLECTOR: (optional) *NXdeflector*

blankerID: (optional) *NXdeflector*

A component for blanking the ion beam or generating pulsed ion beams.

MONOCHROMATOR: (optional) [NXmonochromator](#)

SENSOR: (optional) [NXsensor](#)

ACTUATOR: (optional) [NXactuator](#)

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.

COMPONENT: (optional) [NXcomponent](#)

scan_controller: (optional) [NXscanbox_em](#)

Hypertext Anchors

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

- [/NXbeam_column/ACTUATOR-group](#)
- [/NXbeam_column/APERTURE-group](#)
- [/NXbeam_column/BEAM-group](#)
- [/NXbeam_column/blankerID-group](#)
- [/NXbeam_column/COMPONENT-group](#)
- [/NXbeam_column/DEFLECTOR-group](#)
- [/NXbeam_column/ion_source-group](#)
- [/NXbeam_column/ion_source/brightness-field](#)
- [/NXbeam_column/ion_source/current-field](#)
- [/NXbeam_column/ion_source/description-field](#)
- [/NXbeam_column/ion_source/emitter_type-field](#)
- [/NXbeam_column/ion_source/flux-field](#)
- [/NXbeam_column/ion_source/ion_energy_profile-field](#)
- [/NXbeam_column/ion_source/name-field](#)
- [/NXbeam_column/ion_source/probe-group](#)
- [/NXbeam_column/ion_source/voltage-field](#)
- [/NXbeam_column/LENS_EM-group](#)
- [/NXbeam_column/MONOCHROMATOR-group](#)
- [/NXbeam_column/operation_mode-field](#)
- [/NXbeam_column/scan_controller-group](#)
- [/NXbeam_column/SENSOR-group](#)

NXDL Source:

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

NXimage

Status:

base class, extends [NXobject](#)

Description:

Base class for reporting a set of images representing specializations of NXdata.

The most commonly used scanning methods are supported. That is one-, two-, three-dimensional ROIs discretized using regular Euclidean tilings.

Colloquially, an image is understood as a discretized representation of intensity distribution detected or simulated for some ROI. When discretized with regular Euclidean tilings, the terms pixel and voxel identify the smallest discretization unit. In this case, pixel and voxel are polygonal or polyhedral unit cells respectively of the underlying tiling of the ROI within the reference space. For all other tilings e.g. non-equispaced, the shape and size of pixel and voxel differs. Using the term (image) point is eventually more appropriate when working with such tilings.

Therefore, all docstrings in this base class refer to points (including pixel and voxel i.e. regular tilings).

Point coordinates identify the location of the barycenter.

For images in reciprocal space in practice, complex numbers are encoded via some formatted pair of real values. Typically, fast algorithms for computing Fourier transformations (FFT) are used to encode images in reciprocal (frequency) space. FFT libraries are used for implementing the key functionalities of these mathematical operations. Different libraries use different representations and encoding of the images. Details can be found in the respective sections of the typical FFT libraries documentations:

- [FFTW by M. Frigo and S. G. Johnson](#)
- [Intel MKL by the Intel Co.](#)
- [cuFFT by the NVidia Co.](#)
- [NFFT by the TU Chemnitz group](#) for non-equispaced computations

Users are strongly advised to inspect carefully which specific conventions their library uses to enable storing and modifying the implementation of their code such that the serialized representations as they are detailed here for NeXus match.

It is often the case that several images are combined using processing. In this case, the number of images which are combined into collections is not necessarily the same for each collection. The NXimage base class addresses this logical distinction through the notation of indices_image and indices_group concepts. That is indices_image are always counting from offset in increments of one as each image is its own entity. By contrast, a group may contain no, or several images. Consequently, indices_group are not required to be contiguous.

Symbols:

n_img: Number of images in the stack, for stacks the slowest dimension.

n_k: Number of image points along the slow dimension (k equivalent to z).

n_j: Number of image points along the fast dimension (j equivalent to y).

n_i: Number of image points along the fastest dimension (i equivalent to x).

Groups cited:

[NXdata](#), [NXnote](#), [NXprocess](#), [NXprogram](#)

Structure:

PROCESS: (optional) [NXprocess](#)

Details how NXdata instance were processed from detector readings/raw data.

detector_identifier: (optional) *NX_CHAR*

Link or name of an *NXdetector* instance with which the data were collected.

input: (optional) *NXnote* <=

Resolvable data artifact (e.g. file) from which all values in the *NXdata* instances in this *NXimage* were loaded during parsing.

Possibility to document from which specific other serialized resource as the source pieces of information were processed when using NeXus as a semantic file format to serialize that information differently.

The group in combination with an added field *context* therein adds context.

context: (optional) *NX_CHAR*

Reference to a location inside the artifact that points to the specific group of values that were processed if the artifacts contains several groups of values and thus further resolving of ambiguities is required.

PROGRAM: (optional) *NXprogram*

Program used for processing.

image_1d: (optional) *NXdata*

One-dimensional image.

intensity: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_UNITLESS*}

Intensity for real-valued images as an alternative for real. Magnitude of the image intensity for complex-valued data.

real: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_UNITLESS*}

Real part of the image intensity per point.

imag: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_UNITLESS*}

Imaginary part of the image intensity per point.

complex: (optional) *NX_COMPLEX* (Rank: 1, Dimensions: [n_i]) {units=*NX_UNITLESS*}

Image intensity as a complex number as an alternative to real and imag fields if values are stored as interleaved complex numbers.

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_LENGTH*}

Point coordinate along the fastest dimension.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fastest dimension.

image_2d: (optional) *NXdata*

Two-dimensional image.

intensity: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_j, n_i]) {units=*NX_UNITLESS*}

Intensity for real-valued images as an alternative for real. Magnitude of the image intensity for complex-valued data.

real: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_j, n_i]) {units=*NX_UNITLESS*}

Real part of the image intensity per point.

imag: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_j, n_i]) {units=*NX_UNITLESS*}

Imaginary part of the image intensity per point.

complex: (optional) *NX_COMPLEX* (Rank: 2, Dimensions: [n_j, n_i]) {units=*NX_UNITLESS*}

Image intensity as a complex number as an alternative to real and imag fields if values are stored as interleaved complex numbers.

axis_j: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_j]) {units=*NX_LENGTH*}

Point coordinate along the fast dimension.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fast dimension.

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_LENGTH*}

Point coordinate along the fastest dimension.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fastest dimension.

image_3d: (optional) *NXdata*

Three-dimensional image.

intensity: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [n_k, n_j, n_i]) {units=*NX_UNITLESS*}

Intensity for real-valued images as an alternative for real. Magnitude of the image intensity for complex-valued data.

real: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [n_k, n_j, n_i]) {units=*NX_UNITLESS*}

Real part of the image intensity per point.

imag: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [n_k, n_j, n_i]) {units=*NX_UNITLESS*}

Imaginary part of the image intensity per point.

complex: (optional) *NX_COMPLEX* (Rank: 3, Dimensions: [n_k, n_j, n_i]) {units=*NX_UNITLESS*}

Image intensity as a complex number as an alternative to real and imag fields if values are stored as interleaved complex numbers.

axis_k: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_k]) {units=*NX_LENGTH*}

Point coordinate along the slow dimension.

@long_name: (optional) *NX_CHAR*

Point coordinate along the slow dimension.

axis_j: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_j]) {units=*NX_LENGTH*}

Point coordinate along the fast dimension.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fast dimension.

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_LENGTH*}

Point coordinate along the fastest dimension.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fastest dimension.

stack_1d: (optional) *NXdata*

Collection of one-dimensional images.

intensity: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_img, n_i])
{units=*NX_UNITLESS*}

Intensity for real-valued images as an alternative for real. Magnitude of the image intensity for complex-valued data.

real: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_img, n_i]) {units=*NX_UNITLESS*}

Real part of the image intensity per point.

imag: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_img, n_i]) {units=*NX_UNITLESS*}

Imaginary part of the image intensity per point.

complex: (optional) *NX_COMPLEX* (Rank: 2, Dimensions: [n_img, n_i])
{units=*NX_UNITLESS*}

Image intensity as a complex number as an alternative to real and imag fields if values are stored as interleaved complex numbers.

indices_group: (optional) *NX_INT* (Rank: 1, Dimensions: [n_img]) {units=*NX_UNITLESS*}

Group identifier

@long_name: (optional) *NX_CHAR*

Group identifier

indices_image: (optional) *NX_INT* (Rank: 1, Dimensions: [n_img]) {units=*NX_UNITLESS*}

Image identifier

@long_name: (optional) *NX_CHAR*

Image identifier

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_LENGTH*}

Point coordinate along the fastest dimension.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fastest dimension.

stack_2d: (optional) *NXdata*

Collection of two-dimensional images.

intensity: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [n_img, n_j, n_i])
{units=*NX_UNITLESS*}

Intensity for real-valued images as an alternative for real. Magnitude of the image intensity for complex-valued data.

real: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [n_img, n_j, n_i])
{units=*NX_UNITLESS*}

Real part of the image intensity per point.

imag: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [n_img, n_j, n_i])
 {units=*NX_UNITLESS*}

Imaginary part of the image intensity per point.

complex: (optional) *NX_COMPLEX* (Rank: 3, Dimensions: [n_img, n_j, n_i])
 {units=*NX_UNITLESS*}

Image intensity as a complex number as an alternative to real and imag fields if values are stored as interleaved complex numbers.

indices_group: (optional) *NX_INT* (Rank: 1, Dimensions: [n_img]) {units=*NX_UNITLESS*}

Group identifier

@long_name: (optional) *NX_CHAR*

Group identifier

indices_image: (optional) *NX_INT* (Rank: 1, Dimensions: [n_img]) {units=*NX_UNITLESS*}

Image identifier

@long_name: (optional) *NX_CHAR*

Image identifier.

axis_j: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_j]) {units=*NX_LENGTH*}

Point coordinate along the fast dimension.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fast dimension.

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_LENGTH*}

Point coordinate along the fastest dimension.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fastest dimension.

stack_3d: (optional) *NXdata*

Collection of three-dimensional images.

intensity: (optional) *NX_NUMBER* (Rank: 4, Dimensions: [n_img, n_k, n_j, n_i])
 {units=*NX_UNITLESS*}

Intensity for real-valued images as an alternative for real. Magnitude of the image intensity for complex-valued data.

real: (optional) *NX_NUMBER* (Rank: 4, Dimensions: [n_img, n_k, n_j, n_i])
 {units=*NX_UNITLESS*}

Real part of the image intensity per point.

imag: (optional) *NX_NUMBER* (Rank: 4, Dimensions: [n_img, n_k, n_j, n_i])
 {units=*NX_UNITLESS*}

Imaginary part of the image intensity per point.

complex: (optional) *NX_COMPLEX* (Rank: 4, Dimensions: [n_img, n_k, n_j, n_i])
 {units=*NX_UNITLESS*}

Image intensity as a complex number as an alternative to real and imag fields if values are stored as interleaved complex numbers.

indices_group: (optional) *NX_INT* (Rank: 1, Dimensions: [n_img]) {units=*NX_UNITLESS*}

Group identifier

@long_name: (optional) *NX_CHAR*

Group identifier

indices_image: (optional) *NX_INT* (Rank: 1, Dimensions: [n_img]) {units=*NX_UNITLESS*}

Image identifier

@long_name: (optional) *NX_CHAR*

Image identifier

axis_k: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_k]) {units=*NX_LENGTH*}

Point coordinate along the slow dimension.

@long_name: (optional) *NX_CHAR*

Point coordinate along the slow dimension.

axis_j: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_j]) {units=*NX_LENGTH*}

Point coordinate along the fast dimension.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fast dimension.

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_LENGTH*}

Point coordinate along the fastest dimension.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fastest dimension.

Hypertext Anchors

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

- [/NXimage/image_1d-group](#)
- [/NXimage/image_1d/axis_i-field](#)
- [/NXimage/image_1d/axis_i@long_name-attribute](#)
- [/NXimage/image_1d/complex-field](#)
- [/NXimage/image_1d/imag-field](#)
- [/NXimage/image_1d/intensity-field](#)
- [/NXimage/image_1d/real-field](#)
- [/NXimage/image_2d-group](#)
- [/NXimage/image_2d/axis_i-field](#)
- [/NXimage/image_2d/axis_i@long_name-attribute](#)
- [/NXimage/image_2d/axis_j-field](#)
- [/NXimage/image_2d/axis_j@long_name-attribute](#)
- [/NXimage/image_2d/complex-field](#)

- */NXimage/image_2d/imag-field*
- */NXimage/image_2d/intensity-field*
- */NXimage/image_2d/real-field*
- */NXimage/image_3d-group*
- */NXimage/image_3d/axis_i-field*
- */NXimage/image_3d/axis_i@long_name-attribute*
- */NXimage/image_3d/axis_j-field*
- */NXimage/image_3d/axis_j@long_name-attribute*
- */NXimage/image_3d/axis_k-field*
- */NXimage/image_3d/axis_k@long_name-attribute*
- */NXimage/image_3d/complex-field*
- */NXimage/image_3d/imag-field*
- */NXimage/image_3d/intensity-field*
- */NXimage/image_3d/real-field*
- */NXimage/PROCESS-group*
- */NXimage/PROCESS/detector_identifier-field*
- */NXimage/PROCESS/input-group*
- */NXimage/PROCESS/input/context-field*
- */NXimage/PROCESS/PROGRAM-group*
- */NXimage/stack_1d-group*
- */NXimage/stack_1d/axis_i-field*
- */NXimage/stack_1d/axis_i@long_name-attribute*
- */NXimage/stack_1d/complex-field*
- */NXimage/stack_1d/imag-field*
- */NXimage/stack_1d/indices_group-field*
- */NXimage/stack_1d/indices_group@long_name-attribute*
- */NXimage/stack_1d/indices_image-field*
- */NXimage/stack_1d/indices_image@long_name-attribute*
- */NXimage/stack_1d/intensity-field*
- */NXimage/stack_1d/real-field*
- */NXimage/stack_2d-group*
- */NXimage/stack_2d/axis_i-field*
- */NXimage/stack_2d/axis_i@long_name-attribute*
- */NXimage/stack_2d/axis_j-field*
- */NXimage/stack_2d/axis_j@long_name-attribute*
- */NXimage/stack_2d/complex-field*

- */NXimage/stack_2d/imag-field*
- */NXimage/stack_2d/indices_group-field*
- */NXimage/stack_2d/indices_group@long_name-attribute*
- */NXimage/stack_2d/indices_image-field*
- */NXimage/stack_2d/indices_image@long_name-attribute*
- */NXimage/stack_2d/intensity-field*
- */NXimage/stack_2d/real-field*
- */NXimage/stack_3d-group*
- */NXimage/stack_3d/axis_i-field*
- */NXimage/stack_3d/axis_i@long_name-attribute*
- */NXimage/stack_3d/axis_j-field*
- */NXimage/stack_3d/axis_j@long_name-attribute*
- */NXimage/stack_3d/axis_k-field*
- */NXimage/stack_3d/axis_k@long_name-attribute*
- */NXimage/stack_3d/complex-field*
- */NXimage/stack_3d/imag-field*
- */NXimage/stack_3d/indices_group-field*
- */NXimage/stack_3d/indices_group@long_name-attribute*
- */NXimage/stack_3d/indices_image-field*
- */NXimage/stack_3d/indices_image@long_name-attribute*
- */NXimage/stack_3d/intensity-field*
- */NXimage/stack_3d/real-field*

NXDL Source:

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

NXinsertion_device**Status:**

base class, extends *NXcomponent*

Description:

An insertion device, as used in a synchrotron light source.

Symbols:

No symbol table

Groups cited:

NXdata, *NXgeometry*, *NXoff_geometry*

Structure:

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* <=

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

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/type-field*

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:

NXactuator, *NXaperture*, *NXattenuator*, *NXbeam_stop*, *NXbeam*, *NXbending_magnet*, *NXcapillary*, *NXcollection*, *NXcollimator*, *NXcrystal*, *NXdetector_group*, *NXdetector*, *NXdisk_chopper*, *NXevent_data*, *NXfabrication*, *NXfermi_chopper*, *NXfilter*, *NXflipper*, *NXguide*, *NXhistory*, *NXinsertion_device*, *NXmirror*, *NXmoderator*, *NXmonochromator*, *NXpolarizer*, *NXpositioner*, *NXsensor*, *NXsource*, *NXtransformations*, *NXvelocity_selector*, *NXxraylens*

Structure:

name: (optional) *NX_CHAR*

Name of instrument

@short_name: (optional) *NX_CHAR*

short name for instrument, perhaps the acronym

ACTUATOR: (optional) [*NXactuator*](#)
 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*](#)
 FABRICATION: (optional) [*NXfabrication*](#)
 FERMI_CHOPPER: (optional) [*NXfermi_chopper*](#)
 FILTER: (optional) [*NXfilter*](#)
 FLIPPER: (optional) [*NXflipper*](#)
 GUIDE: (optional) [*NXguide*](#)
 HISTORY: (optional) [*NXhistory*](#)
 INSERTION_DEVICE: (optional) [*NXinsertion_device*](#)
 MIRROR: (optional) [*NXmirror*](#)
 MODERATOR: (optional) [*NXmoderator*](#)
 MONOCHROMATOR: (optional) [*NXmonochromator*](#)
 POLARIZER: (optional) [*NXpolarizer*](#)
 POSITIONER: (optional) [*NXpositioner*](#)
 SENSOR: (optional) [*NXsensor*](#)
 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/ACTUATOR-group*](#)
- [*/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/FABRICATION-group*
- */NXinstrument/FERMI_CHOPPER-group*
- */NXinstrument/FILTER-group*
- */NXinstrument/FLIPPER-group*
- */NXinstrument/GUIDE-group*
- */NXinstrument/HISTORY-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/SENSOR-group*
- */NXinstrument/SOURCE-group*
- */NXinstrument/VELOCITY_SELECTOR-group*
- */NXinstrument/XRAYLENS-group*

NXDL Source:

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

NXinstrument_em

Status:

base class, extends [NXinstrument](#)

Description:

Base class for instrument-related details of a real or simulated electron microscope.

For collecting data and experiments which are simulations of an electron microscope (or such session) use the [NXem](#) application definition and the [NXevent_data_em](#) groups it provides.

This base class implements the concept of [NXem](#) whereby (meta)data are distinguished whether these typically change during a session (dynamic) or not (static metadata). This design allows to store e.g. hardware related concepts only once instead of demanding that each image or spectrum from the session needs to be stored also with the static metadata.

Symbols:

No symbol table

Groups cited:

[NXactuator](#), [NXcomponent](#), [NXdetector](#), [NXbeam_column](#), [NXfabrication](#), [NXbeam_column](#), [NXmanipulator](#), [NXoptical_system_em](#), [NXpump](#), [NXsensor](#)

Structure:

name: (optional) [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.

type: (recommended) [NX_CHAR](#)

Different types of electron microscopes exist:

- sem, a scanning electron microscope without focused-ion beam capabilities
- fib, a scanning electron microscope with focused-ion beam capabilities irrespective whether these were used or not
- tem, a transmission electron microscope

NXem is one joint data model that can be used to document research that is performed with several of these types of microscopes (SEM, TEM, or FIB). The NXem data model stresses that these types of instruments despite having several differences are still all electron beamlines with which to probe electron and/or ion matter interaction and in fact in practice have many similarities in how they are used, the components, they contain, etc.

This field can be used in research data management systems for enabling a categorization or tagging of experiments without having to analyze if groups like NXbeam_column are present (which would indicate type is fib) or if certain lens configurations or instrument models are used which suggests the microscope is a scanning (sem) or transmission electron microscope (tem):

Any of these values: `sem | fib | tem`

FABRICATION: (optional) [NXfabrication](#) <=

EBEAM_COLUMN: (optional) [NXbeam_column](#)

IBEAM_COLUMN: (optional) [NXbeam_column](#)

OPTICAL_SYSTEM_EM: (optional) [NXoptical_system_em](#)

DETECTOR: (optional) [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.

stageID: (optional) [NXmanipulator](#)

Stages in an electron microscope are multi-functional devices.

Stages enable experimentalists the application of controlled external stimuli on the specimen. Modern stages realize a hierarchy of components. A multi-axial tilt rotation holder is a good example where the control of each degree of freedom is technically implemented via providing instances of e.g. [NXpositioner](#) or [NXactuator](#) that achieve the rotating and positioning of the specimen.

The physical process of mounting a specimen on a stage in practice often comes with an own hierarchy of fixtures to bridge e.g. length scales technically. An example from atom probe microscopy is that researchers may work with wire samples which are clipped into a larger fixing unit to enable careful specimen handling. Alternatively, a microtip is a silicon post upon which e.g. an atom probe specimen is mounted. Multiple of such microtips are then grouped into a microtip array to conveniently enable loading of multiple specimens into the instrument with fewer operations. There are further scenarios typically encountered related to mounting and locating specimens inside an electron microscope, a few examples follow:

- A nanoparticle on a copper grid. The copper grid is the holder. This grid itself is fixed to a 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.
- For in-situ experiments with e.g. chips with read-out electronics as actuators, the chips are again placed in a larger unit. A typical example are in-situ experiments using e.g. the tools of [Protochips](#).
- 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.

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

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

alias: (optional) *NX_CHAR*

Free-text field to give a term how that a stage_lab at this level of the stage_lab hierarchy is commonly referred to. Examples could be stub, puck, carousel, microtip, clip, holder, etc.

tilt1: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

The interpretation of this tilt1 value can be contextualized via the comment attribute. However, it is better to describe the reference frame in which the tilt is defined explicitly using instances of *NXtransformations* and respective instances of *NXcoordinate_system*. Especially when this NXinstrument_em base class is used in an application definition like NXem.

@comment: (optional) *NX_CHAR*

Discouraged free-text field to provide details about how to interpret tilt1.

tilt2: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

The interpretation of this tilt2 value can be contextualized via the comment attribute. However, it is better to describe the reference frame in which the tilt is defined explicitly using instances of *NXtransformations* and respective instances of *NXcoordinate_system*. Especially when this NXinstrument_em base class is used in an application definition like NXem.

@comment: (optional) *NX_CHAR*

Discouraged free-text field to provide details about how to interpret tilt2.

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

The interpretation of this rotation value can be contextualized via the comment attribute. However, it is better to describe the reference frame in which the rotation is defined explicitly using instances of *NXtransformations* and respective instances of *NXcoordinate_system*. Especially when this NXinstrument_em base class is used in an application definition like NXem.

@comment: (optional) *NX_CHAR*

Discouraged free-text field to provide details about how to interpret rotation.

position: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

The interpretation of these position values can be contextualized via the comment attribute. However, it is better to describe the reference frame in which the position values are defined explicitly using instances of *NXtransformations* and respective instances of *NXcoordinate_system*. Especially when this NXinstrument_em base class is used in an application definition like NXem.

nanoprobeID: (optional) *NXmanipulator*

In contrast to the stage, the nanoprobe is an additional manipulator that is a specifically frequently found component of FIB/SEM instruments. A nanoprobe is used to pick up and relocated portions of the specimen that have been cut off during site-specific lift-outs and specimen preparation.

gas_injector: (optional) *NXcomponent*

Gas injection systems (GIS) are components of microscopes that are equipped with focused-ion beam capabilities. The component is used to introduce reactive neutral gases to the sample surface for enhanced etching, preferential etching, or material deposition.

PUMP: (optional) *NXpump*

SENSOR: (optional) *NXsensor* <=

ACTUATOR: (optional) *NXactuator* <=

Hypertext Anchors

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

- */NXinstrument_em/ACTUATOR-group*
- */NXinstrument_em/DETECTOR-group*
- */NXinstrument_em/EBEAM_COLUMN-group*
- */NXinstrument_em/FABRICATION-group*
- */NXinstrument_em/gas_injector-group*
- */NXinstrument_em/IBEAM_COLUMN-group*
- */NXinstrument_em/location-field*
- */NXinstrument_em/name-field*
- */NXinstrument_em/nanoprobeID-group*
- */NXinstrument_em/OPTICAL_SYSTEM_EM-group*
- */NXinstrument_em/PUMP-group*
- */NXinstrument_em/SENSOR-group*
- */NXinstrument_em/stageID-group*
- */NXinstrument_em/stageID/alias-field*
- */NXinstrument_em/stageID/design-field*
- */NXinstrument_em/stageID/position-field*
- */NXinstrument_em/stageID/rotation-field*
- */NXinstrument_em/stageID/rotation@comment-attribute*
- */NXinstrument_em/stageID/tilt1-field*
- */NXinstrument_em/stageID/tilt1@comment-attribute*
- */NXinstrument_em/stageID/tilt2-field*
- */NXinstrument_em/stageID/tilt2@comment-attribute*
- */NXinstrument_em/type-field*

NXDL Source:

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

NXinteraction_volume_em

Status:

base class, extends [NXobject](#)

Description:

Base class to describe the volume of interaction for particle-matter interaction.

Computer models like Monte Carlo or molecular dynamics / electron- or ion-beam interaction simulations can be used to qualify and (or) quantify the shape of the interaction volume. Results of such simulations can be summary statistics or single-particle-resolved sets of trajectories.

Explicit or implicit descriptions of the geometry of this interaction volume 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 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 flux through an imaginary control surface (the iso-surface) 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.](#)
- [J. F. Ziegler 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_volume_em/DATA-group](#)
- [/NXinteraction_volume_em/PROCESS-group](#)

NXDL Source:

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

NXion

Status:

base class, extends [NXatom](#)

Description:

Base class for documenting the set of atoms of a (molecular) ion.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ivc_max: 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:

id: (optional) *NX_CHAR*

A unique identifier whereby such an ion can be referred to via the service offered as described in id_type.

id_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.

nuclide_hash: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_ivc_max]) {units=*NX_UNITLESS*}

Vector of nuclide hash values.

Individual hash values H is $H = Z + N \cdot 256$ with Z encode the number of protons Z and the number of neutrons N of each nuclide respectively. Z and N have to be 8-bit unsigned integers.

The array is sorted in decreasing order. For the rationale behind this see M. Kühbach et al. (2021)

nuclide_list: (optional) *NX_UINT* (Rank: 2, Dimensions: [n_ivc_max, 2]) {units=*NX_UNITLESS*}

Table which decodes the entries in nuclide_hash into a human-readable matrix of instances. The first column specifies the nuclide mass number, i.e. using the hashvalues from the isotope_vector this is $Z + N$ or 0. The value 0 documents that no isotope-specific information about the element encoded is relevant. The second row specifies the number of protons Z or 0. The value 0 in this case documents a placeholder or that no element-specific information is relevant. Taking a carbon-14 nuclide as an example the mass number is 14. That is encoded as a value pair (14, 6) as one row of the table.

Therefore, this notation is the typical superscribed nuclide mass number and subscripted number of protons element notation e.g. ^{14}C . The array is stored matching the order of nuclide_hash.

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

Associated lower (mqmin) and upper (mqmax) bounds of the mass-to-charge-state ratio interval(s) [mqmin, mqmax] (boundaries inclusive). This field is primarily of interest for documenting *NXprocess* steps of indexing a ToF/mass-to-charge state histogram.

Hypertext Anchors

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

- [*/NXion/id-field*](#)
- [*/NXion/id_type-field*](#)
- [*/NXion/ion_type-field*](#)
- [*/NXion/mass_to_charge_range-field*](#)
- [*/NXion/nuclide_hash-field*](#)
- [*/NXion/nuclide_list-field*](#)

NXDL Source:

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

NXlens_em

Status:

base class, extends [*NXcomponent*](#)

Description:

Base class for 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: Scanning Electron Microscopy
- P. Hawkes: Magnetic Electron Lenses
- Y. Liao: Practical Electron Microscopy and Database

Symbols:

No symbol table

Groups cited:

none

Structure:

name: (optional) [*NX_CHAR*](#) <=

Name of the lens.

description: (optional) [*NX_CHAR*](#) <=

Ideally, use instances of **identifierNAME** to point to a resource that provides further details.

If such a resource does not exist or should not be used, use this free text, although it is not recommended.

power_setting: (optional) [*NX_CHAR_OR_NUMBER*](#) {units=[*NX_ANY*](#)}

Descriptor for the lens excitation when the exact technical details are unknown or not directly controllable as the control software of the microscope does not enable or was not configured to display these values for users.

Although this value does not document the exact physical voltage or excitation, it can still give useful context to reproduce the lens setting, provided a properly working instrument and software sets the lens into a similar state to the technical level possible when no more information is available physically or accessible legally.

mode: (optional) *NX_CHAR*

Descriptor for the operation mode of the lens when other details are not directly controllable as the control software of the microscope does not enable or is not configured to display these values.

Like value, the mode can only be interpreted for a specific microscope but can still be useful to guide users as to how to repeat the measurement.

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

Excitation voltage of the lens.

For dipoles it is a single number. For higher order multipoles, 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-order multipoles, it is an array.

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
- dodecapole

Hypertext Anchors

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

- [*/NXlens_em/current-field*](#)
- [*/NXlens_em/description-field*](#)
- [*/NXlens_em mode-field*](#)
- [*/NXlens_em/name-field*](#)
- [*/NXlens_em/power_setting-field*](#)
- [*/NXlens_em/type-field*](#)
- [*/NXlens_em/voltage-field*](#)

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/base_classes/NXlens_em.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:

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*

NXDL Source:

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

NXmanipulator**Status:**

base class, extends *NXcomponent*

Description:

Base class to describe the use of manipulators and sample stages.

Symbols:

No symbol table

Groups cited:

NXactuator, *NXlog*, *NXpid_controller*, *NXpositioner*, *NXsensor*

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.

cryostat: (optional) *NXactuator*

Cryostat for cooling the sample (and, potentially, the whole manipulator).

actuation_target: (optional) *NX_CHAR* <=

Obligatory value: `temperature`

PID_CONTROLLER: (optional) *NXpid_controller* <=

setpoint: (optional) *NX_FLOAT* {units=*NX_TEMPERATURE*} <=

In case of a fixed or averaged cooling temperature, this is the scalar temperature setpoint. It can also be a 1D array of temperature setpoints (without time stamps).

setpoint_log: (optional) *NXlog*

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

In the case of an experiment in which the temperature is changed and the setpoints are recorded with time stamps, this is an array of temperature setpoints.

temperature_sensor: (optional) *NXsensor*

Temperature sensor measuring the sample temperature.

measurement: (optional) *NX_CHAR* <=

Obligatory value: `temperature`

value: (optional) *NX_FLOAT* {units=*NX_TEMPERATURE*} <=

In case of a single or averaged temperature measurement, this is the scalar temperature measured by the sample temperature sensor. It can also be a 1D array of measured temperatures (without time stamps).

value_log: (optional) *NXlog* <=

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

In the case of an experiment in which the temperature changes and is recorded with time stamps, this is an array of length m of temperatures.

sample_heater: (optional) *NXactuator*

Device to heat the sample.

actuation_target: (optional) *NX_CHAR* <=

Obligatory value: `temperature`

output_heater_power: (optional) *NX_FLOAT* {units=*NX_POWER*}

In case of a fixed or averaged heating power, this is the scalar heater power. It can also be a 1D array of heater powers (without time stamps).

output_heater_power_log: (optional) *NXlog*

value: (optional) *NX_FLOAT* {units=*NX_POWER*}

In the case of an experiment in which the heater power is changed and recorded with time stamps, this is an array of length m of temperature set-points.

PID_CONTROLLER: (optional) *NXpid_controller* <=

setpoint: (optional) *NX_FLOAT* {units=*NX_TEMPERATURE*} <=

In case of a fixed or averaged temperature, this is the scalar temperature setpoint. It can also be a 1D array of temperature setpoints (without time stamps).

setpoint_log: (optional) *NXlog*

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

In the case of an experiment in which the temperature is changed and the setpoints are recorded with time stamps, this is an array of length m of temperature setpoints.

drain_current_ammeter: (optional) *NXsensor*

Ammeter measuring the drain current of the sample and sample holder.

measurement: (optional) *NX_CHAR* <=

Obligatory value: `current`

value: (optional) *NX_FLOAT* {units=*NX_CURRENT*} <=

In case of a single or averaged drain current measurement, this is the scalar drain current measured between the sample and sample holder. It can also be an 1D array of measured currents (without time stamps).

value_log: (optional) *NXlog* <=

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

In the case of an experiment in which the current changes and is recorded with time stamps, this is an array of length m of currents.

sample_bias_potentiostat: (optional) *NXactuator*

Actuator applying a voltage between sample holder and sample.

actuation_target: (optional) *NX_CHAR* <=

Obligatory value: *voltage*

PID_CONTROLLER: (optional) *NXpid_controller* <=

setpoint: (optional) *NX_FLOAT* {units=*NX_VOLTAGE*} <=

In case of a fixed or averaged applied bias, this is the scalar voltage applied between sample and sample holder. It can also be an 1D array of voltage setpoints (without time stamps).

setpoint_log: (optional) *NXlog*

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

In the case of an experiment in which the bias is changed and the setpoints are recorded with time stamps, this is an array of length m of voltage setpoints.

sample_bias_voltmeter: (optional) *NXsensor*

Sensor measuring the voltage applied to sample and sample holder.

measurement: (optional) *NX_CHAR* <=

Obligatory value: *voltage*

value: (optional) *NX_FLOAT* {units=*NX_VOLTAGE*} <=

In case of a single or averaged bias measurement, this is the scalar voltage measured between sample and sample holder. It can also be an 1D array of measured voltages (without time stamps).

value_log: (optional) *NXlog* <=

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

In the case of an experiment in which the bias changes and is recorded with time stamps, this is an array of length m of voltages.

ACTUATOR: (optional) *NXactuator*

Any additional actuator on the manipulator used to control an external condition.

SENSOR: (optional) *NXsensor*

Any additional sensors on the manipulator used to monitor an external condition.

POSITIONER: (optional) *NXpositioner*

Class to describe the motors that are used in the manipulator.

Hypertext Anchors

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

- */NXmanipulator/ACTUATOR-group*
- */NXmanipulator/cryostat-group*

- */NXmanipulator/cryostat/actuation_target-field*
- */NXmanipulator/cryostat/PID_CONTROLLER-group*
- */NXmanipulator/cryostat/PID_CONTROLLER/setpoint-field*
- */NXmanipulator/cryostat/PID_CONTROLLER/setpoint_log-group*
- */NXmanipulator/cryostat/PID_CONTROLLER/setpoint_log/value-field*
- */NXmanipulator/description-field*
- */NXmanipulator/drain_current_ammeter-group*
- */NXmanipulator/drain_current_ammeter/measurement-field*
- */NXmanipulator/drain_current_ammeter/value-field*
- */NXmanipulator/drain_current_ammeter/value_log-group*
- */NXmanipulator/drain_current_ammeter/value_log/value-field*
- */NXmanipulator/name-field*
- */NXmanipulator/POSITIONER-group*
- */NXmanipulator/sample_bias_potentiostat-group*
- */NXmanipulator/sample_bias_potentiostat/actuation_target-field*
- */NXmanipulator/sample_bias_potentiostat/PID_CONTROLLER-group*
- */NXmanipulator/sample_bias_potentiostat/PID_CONTROLLER/setpoint-field*
- */NXmanipulator/sample_bias_potentiostat/PID_CONTROLLER/setpoint_log-group*
- */NXmanipulator/sample_bias_potentiostat/PID_CONTROLLER/setpoint_log/value-field*
- */NXmanipulator/sample_bias_voltmeter-group*
- */NXmanipulator/sample_bias_voltmeter/measurement-field*
- */NXmanipulator/sample_bias_voltmeter/value-field*
- */NXmanipulator/sample_bias_voltmeter/value_log-group*
- */NXmanipulator/sample_bias_voltmeter/value_log/value-field*
- */NXmanipulator/sample_heater-group*
- */NXmanipulator/sample_heater/actuation_target-field*
- */NXmanipulator/sample_heater/output_heater_power-field*
- */NXmanipulator/sample_heater/output_heater_power_log-group*
- */NXmanipulator/sample_heater/output_heater_power_log/value-field*
- */NXmanipulator/sample_heater/PID_CONTROLLER-group*
- */NXmanipulator/sample_heater/PID_CONTROLLER/setpoint-field*
- */NXmanipulator/sample_heater/PID_CONTROLLER/setpoint_log-group*
- */NXmanipulator/sample_heater/PID_CONTROLLER/setpoint_log/value-field*
- */NXmanipulator/SENSOR-group*
- */NXmanipulator/temperature_sensor-group*
- */NXmanipulator/temperature_sensor/measurement-field*

- `/NXmanipulator/temperature_sensor/value-field`
- `/NXmanipulator/temperature_sensor/value_log-group`
- `/NXmanipulator/temperature_sensor/value_log/value-field`
- `/NXmanipulator/type-field`

NXDL Source:

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

NXmirror**Status:**

base class, extends `NXcomponent`

Description:

A beamline mirror or supermirror.

Symbols:

No symbol table

Groups cited:

`NXdata`, `NXgeometry`, `NXoff_geometry`, `NXshape`

Structure:

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`

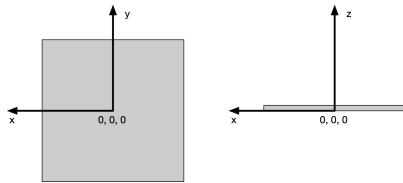
substrate_roughness: (optional) `NX_FLOAT {units=NX_LENGTH}`

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* <=

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

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/type-field*

NXDL Source:

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

NXmoderator**Status:**

base class, extends *NXcomponent*

Description:

A neutron moderator

Symbols:

No symbol table

Groups cited:

NXdata, *NXgeometry*, *NXlog*, *NXoff_geometry*

Structure:

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*

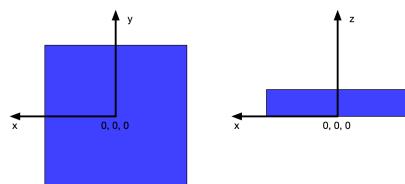
Any of these values: Gd | Cd

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

average/nominal moderator temperature

depends_on: (optional) *NX_CHAR* <=

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

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/type-field](#)

NXDL Source:

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

NXmonitor

Status:

base class, extends [NXcomponent](#)

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](#)

Structure:

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*}

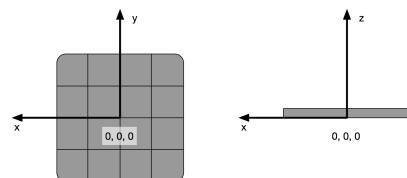
Proportion of incident beam sampled by the monitor (0<x<1)

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* <=

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 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

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/type-field*

NXDL Source:

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

NXmonochromator

Status:

base class, extends *NXcomponent*

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:

NXaperture, NXcrystal, NXdata, NXgeometry, NXgrating, NXoff_geometry, NXvelocity_selector

Structure:

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

energy_dispersion: (optional) *NX_FLOAT* {units=eV/mm}

Energy dispersion at the exit slit.

wavelength_dispersion: (optional) *NX_FLOAT* {units=nm/mm}

Wavelength dispersion at the exit slit.

depends_on: (optional) *NX_CHAR* <=

entrance_slit: (optional) *NXaperture*

Size, position and shape of the entrance slit of the monochromator.

exit_slit: (optional) *NXaperture*

Size, position and shape of the exit slit of the monochromator.

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

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_dispersion-field*
- */NXmonochromator/energy_error-field*
- */NXmonochromator/energy_errors-field*
- */NXmonochromator/entrance_slit-group*
- */NXmonochromator/exit_slit-group*
- */NXmonochromator/geometry-group*
- */NXmonochromator/GRATING-group*
- */NXmonochromator/OFF_GEOMETRY-group*
- */NXmonochromator/VELOCITY_SELECTOR-group*
- */NXmonochromator/wavelength-field*
- */NXmonochromator/wavelength_dispersion-field*
- */NXmonochromator/wavelength_error-field*
- */NXmonochromator/wavelength_errors-field*

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:

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

identifierNAME: (optional) *NX_CHAR*

Identifier of the resource if that resource that has been serialized.

For example, the identifier to a resource in another database.

checksum: (optional) *NX_CHAR*

Value of the hash that is obtained when running algorithm on the content of the resource referred to by **identifierNAME**.

algorithm: (optional) *NX_CHAR*

Name of the algorithm whereby the **checksum** was computed.

Examples: md5, sha256

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/algorithm-field*
- */NXnote/author-field*
- */NXnote/checksum-field*
- */NXnote/data-field*
- */NXnote/date-field*
- */NXnote/description-field*
- */NXnote/file_name-field*
- */NXnote/identifierNAME-field*
- */NXnote/sequence_index-field*

- */NXnote/type-field*

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. The groups and fields contained within this file are allowed to be present in any derived base class.

If nameType=“partial”, the placeholders (e.g., FIELDNAME or GROUPNAME) can be replaced by the name of any object (field or group, respectively) that exists within the same group.

Symbols:

No symbol table

Groups cited:

NXcollection, NXdata, NXlog, NXnote, NXparameters

Structure:**@default:** (optional) *NX_CHAR*

Declares which child group contains a path leading to a *NXdata* group or a group using a base class extending *NXdata*.

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.

FIELDNAME_set: (optional) *NX_NUMBER*

Target values of FIELDNAME.

FIELDNAME_errors: (optional) *NX_NUMBER*

Uncertainties of FIELDNAME values.

FIELDNAME_weights: (optional) *NX_NUMBER*

Weights of FIELDNAME values.

FIELDNAME_mask: (optional) *NX_BOOLEAN*

Boolean mask of FIELDNAME values. The value is masked if set to 1.

identifierNAME: (optional) *NX_CHAR*

An identifier for a (persistent) resource.

An identifier, provided by some authority, that has been assigned to an object described by this **NXObject**. To be useful, the identifier must not be reassigned to a different real-world object. It is typical for there to be some mechanism to resolve an identifier, obtaining metadata about the object. Identifiers for which some guarantees exist regarding this resolution process are called persistent identifiers. Persistent identifiers are also known as PIDs.

@type: (optional) *NX_CHAR*

The type of identifier used.

It is recommended to use the most specific type when describing the identifier.

For example, all IGSNs (see below) are DOIs and all DOIs are Handles; however, an IGSN should have type IGSN (and not DOI or Hdl). Similarly, an ARK, Purl, ORCID and ROR identifiers should have their corresponding types and should not use the more generic URL identifier.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- ARK: Archival Resource Key. An ARK is a Uniform Resource Identifier (URI) designed to support long-term access to a variety of information objects. Syntax: [https://NMA/ark:/NAAN/Name{\[\]}Qualifier](https://NMA/ark:/NAAN/Name{[]}Qualifier). Brackets indicate optional elements. Example: <https://example.org/ark:/12345/abcde>
- DOI: Digital Object Identifier. A DOI is a unique alphanumeric string used to identify digital content. It consists of a prefix and a suffix, separated by a slash. Syntax: 10.XXXX/XXXXXX Example: 10.1107/S1600576714027575
- Hdl: A handle is a unique identifier that consists of a prefix indicating the naming authority and a suffix representing the local name of a resource. A handle is a unique identifier used to facilitate the identification and management of digital objects. It is composed of a prefix that indicates the naming authority and a suffix that specifies the resource's local name. This refers specifically to an ID in the Handle system operated by the Corporation for National Research Initiatives (CNRI). Syntax: prefix/identifier Example: 123456789/abc123
- IGSN: International Generic Sample Number. The IGSN is a unique identifier assigned to a specific sample or specimen in the context of scientific research. Since 2021, IGSNs are issued by DataCite, meaning that there are now DataCite-issued DOIs for all IGSNs, including those historical IGSNs issued beforehand. Therefore, the syntax is the same as for DOIs. Syntax: 10.XXXX/XXXXXX Example: 10.1107/S1600576714027575
- ISNI: ISNI is an ISO standard to uniquely identify individuals and organizations involved in creative work, including pseudonyms and other public personas. An ISNI-ID is made up of 16 digits, the last character being a check character. The check character may be either a decimal digit or the character “X”. A URL can be generated from the ISNI ID by combining it with the prefix <https://isni.org/isni/>, resulting in <https://isni.org/isni/{ISNI-ID}>. Syntax: 16 base-10 digits stored without any spaces. Example: 0000000121032683
- ISSN: International Standard Serial Number An ISSN is an 8-digit unique identifier used to distinguish a serial publication, whether in print or electronic form. The last character (a digit or ‘X’) serves as a check character, making the ISSN uniquely defined by its first seven digits. Syntax: NNNN-NNNC, where N a decimal digit character (i.e., in the set {0,1,2,...,9}), and C is in {0,1,2,...,9,X} Example: 1234-5678 or 1234-567X
- ISSN-L: Linking ISSN The linking ISSN, or ISSN-L, is a specific ISSN that groups the different media of the same serial publication. Syntax: NNNN-NNNC, where N a decimal digit character (i.e., in the set {0,1,2,...,9}), and C is in {0,1,2,...,9,X} Example: 1234-5678 or 1234-567X
- ORCID: Open Researcher and Contributor identifier. ORCID provides a free and persistent identifier that uniquely distinguishes authors and contributors in sci-

entific research. Syntax: <https://orcid.org/XXXX-XXXX-XXXX-XXXX> Example: <https://orcid.org/0000-0002-1825-0097>

- PURL: Persistent Uniform Resource Locator. A Persistent Uniform Resource Locator (PURL) is a type of URL designed to provide a stable, long-term reference to a web resource by using a resolver to redirect users to the resource's current location, even if it moves over time. A PURL has three parts: (1) a protocol, (2) a resolver address, and (3) a name. Syntax: <https://purl.org/foo/bar> Example: <https://purl.org/dc/elements/1.1/title>
- ROR: Research Organization Registry A ROR ID is a globally unique identifier for research organizations, enabling unambiguous linking of institutions across systems. Syntax: [https://ror.org/\[ROR-ID\]](https://ror.org/[ROR-ID]) Example: <https://ror.org/052gg0110>
- URL: Uniform Resource Locator Also known as web address, a URL is the address used to access a resource on the internet, specifying its location and the protocol to retrieve it. Syntax: scheme://domain:port/path?query_string#fragment_id Example: <https://www.example.com/about>
- URN: Uniform Resource Name A URN is a unique, persistent identifier for a resource regardless of where it is stored. It is recommended that identifiers with more specific type attribute (such as DOI or ISSN) values should not be stored as a URN, even when this is valid. As an example, the URN doi:10.1107/S1600576714027575 is a valid URN-based representation for the DOI 10.1107/S1600576714027575, but it is recommended to use type="DOI" in this case. Syntax: urn:<namespace>:<namespace-specific-string>. The leading urn: sequence is case-insensitive. Example: urn:isbn:00000000000000

COLLECTION: (optional) *NXcollection*

DATA: (optional) *NXdata*

LOG: (optional) *NXlog*

NOTE: (optional) *NXnote*

PARAMETERS: (optional) *NXparameters*

GROUPNAME_log: (optional) *NXlog*

NXlog group containing logged values of GROUPNAME.

Hypertext Anchors

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

- /*NXObject*/COLLECTION-group
- /*NXObject*/DATA-group
- /*NXObject*/FIELDNAME_errors-field
- /*NXObject*/FIELDNAME_mask-field
- /*NXObject*/FIELDNAME_set-field
- /*NXObject*/FIELDNAME_weights-field
- /*NXObject*/GROUPNAME_log-group
- /*NXObject*/identifierNAME-field
- /*NXObject*/identifierNAME@type-attribute

- */NXobject/LOG-group*
- */NXobject/NOTE-group*
- */NXobject/PARAMETERS-group*
- */NXobject@default-attribute*

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 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:

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*](#)

NXDL Source:

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

NXoptical_system_em

Status:

base class, extends [*NXObject*](#)

Description:

Base class for qualifying an electron optical system.

Symbols:

No symbol table

Groups cited:

none

Structure:

camera_length: (optional) [*NX_NUMBER*](#) {units=[*NX_LENGTH*](#)}

Distance which is present between the specimen surface and the detector plane.

This concept is related to term [Camera Length](#) of the EMglossary standard.

magnification: (optional) [*NX_NUMBER*](#) {units=[*NX_DIMENSIONLESS*](#)}

The factor of enlargement of the apparent size, not the physical size, of an object.

defocus: (optional) [*NX_NUMBER*](#) {units=[*NX_LENGTH*](#)}

The defocus aberration constant (oftentimes referred to as c_1_0). See respective details in [*NXaberration*](#) class instances.

semi_convergence_angle: (optional) [*NX_NUMBER*](#) {units=[*NX_ANGLE*](#)}

The angle which is given by the semi-opening angle of the cone in a convergent beam.

This concept is related to term [Convergence Angle](#) of the EMglossary standard.

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_NUMBER*](#) {units=[*NX_LENGTH*](#)}

Distance which is determined along the optical axis within the column from (1) the lower end of the final optical element between the source and the specimen stage; to (2) the point where the beam is focused.

This concept is related to term [Working Distance](#) of the EMglossary standard.

probe: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [2])

Geometry of the cross-section formed when the primary beam shines onto the specimen surface. Reported as length of the semiaxes of the ellipsoidal cross-section with semiaxes values sorted by decreasing length.

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

Electrical current which arrives at the specimen.

This concept is related to term [Probe Current](#) of the EMglossary standard.

dose_management: (optional) *NX_CHAR*

Specify further details how incipient electron or ion dose was quantified (using beam_current, probe_current).

[Reference](#) discusses an approach for (electron) dose monitoring in an electron microscope.

The unit of the nominal dose rate is e-/(angstrom²*s).

dose_rate: (optional) *NX_NUMBER* {units=1/(angstrom²*s)}

Nominal dose rate.

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

In the process of passing through an *NXlens_em* electrons are typically accelerated on a helical path about the optical axis. This causes an image rotation whose strength is affected by the magnification.

Microscopes may be equipped with compensation methods (implemented in hardware or software) that reduce but not necessarily eliminate this rotation.

See [L. Reimer](#) for details.

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

Distance which lies between the principal plane of the lens and the focal point along the optical axis.

This concept is related to term [Focal Length](#) of the EMglossary standard.

tilt_correction: (optional) *NX_BOOLEAN*

Details about an imaging setting used during acquisition to correct perspective distortion when imaging a tilted surface or cross section.

This concept is related to term [Tilt Correction](#) of the EMglossary standard.

dynamic_focus_correction: (optional) *NX_BOOLEAN*

Details about a dynamic focus correction used.

This concept is related to term [Dynamic Focus Correction](#) of the EMglossary standard.

dynamic_refocusing: (optional) *NX_CHAR*

Details about a workflow used to keep the specimen in focus by automatic means.

This concept is related to term [Dynamic Refocusing](#) of the EMglossary standard.

Hypertext Anchors

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

- [*/NXoptical_system_em/camera_length-field*](#)
- [*/NXoptical_system_em/defocus-field*](#)
- [*/NXoptical_system_em/dose_management-field*](#)
- [*/NXoptical_system_em/dose_rate-field*](#)
- [*/NXoptical_system_em/dynamic_focus_correction-field*](#)
- [*/NXoptical_system_em/dynamic_refocusing-field*](#)
- [*/NXoptical_system_em/field_of_view-field*](#)
- [*/NXoptical_system_em/focal_length-field*](#)
- [*/NXoptical_system_em/magnification-field*](#)
- [*/NXoptical_system_em/probe-field*](#)
- [*/NXoptical_system_em/probe_current-field*](#)
- [*/NXoptical_system_em/rotation-field*](#)
- [*/NXoptical_system_em/semi_convergence_angle-field*](#)
- [*/NXoptical_system_em/tilt_correction-field*](#)
- [*/NXoptical_system_em/working_distance-field*](#)

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/base_classes/NXoptical_system_em.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:

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 [$x' \cdot x$, $x' \cdot y$, $x' \cdot z$, $y' \cdot x$, $y' \cdot y$, $y' \cdot z$] where “dot” is the scalar dot product (cosine of the angle between the unit vectors). The unit vectors in both the local and reference coordinates are right-handed and orthonormal.

The pair of groups NXtranslation and NXorientation together describe the position of a component.

GEOMETRY: (optional) *NXgeometry*

Link to another object if we are using relative positioning, else absent

Hypertext Anchors

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

- */NXorientation/GEOMETRY-group*
- */NXorientation/value-field*

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:

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*

NXDL Source:

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

NXpdb

Status:

base class, extends *NXObject*

Description:

A NeXus transliteration of a PDB file, to be validated only as a PDB rather than in NeXus.

Use *NXpdb* to incorporate the information in an arbitrary PDB into a NeXus file.

The main suggestion is to use this as a container class for a PDB entry to describe a sample in NXsample, but it may be more appropriate to place this higher in the hierarchy, say in NXentry.

The structure has to follow the structure of a PDB with each PDB data block mapped to a NeXus group of class NXpdb, using a lowercase version of the data block name as the name of the NeXus group, each PDB category in that data block mapped to a NeXus group of class NXpdb and with each PDB column mapped to a NeXus field. Each column in a looped PDB category should always be presented as a 1-dimensional array. The columns in an unlooped PDB category should be presented as scalar values. If a PDB category specifies particular units for columns, the same units should be used for the corresponding fields.

A PDB entry is unambiguous when all information is carried as text. All text data should be presented as quoted strings, with the quote marks except for the null values “.” or “?”

For clarity in NXpdb form, numeric data may be presented using the numeric types specified in the mmCIF dictionary. In that case, if a PDB null value, “.” or “?”, is contained in a numeric column, the IEEE nan should be used for “?” and the IEEE inf should be used for “.”.

An arbitrary DDL2 CIF file can be represented in NeXus using NXpdb. However, if save frames are required, an NXpdb_class attribute with the value “CBF_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
@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
#
```

(continues on next page)

(continued from previous page)

```

_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
_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/NXpeak.nxdl.xml

NXpeak**Status:**

base class, extends *NXObject*

Description:

Base class for describing a peak, its functional form, and support values (i.e., the discretization (points) at which the function has been evaluated).

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

dimRank: Rank of the dependent and independent data arrays (for multidimensional/multivariate fit.)

Groups cited:

NXdata

Structure:

label: (optional) *NX_CHAR*

Human-readable label which specifies which concept/entity the peak represents/identifies.

total_area: (optional) *NX_NUMBER* {units=*NX_ANY*}

Total area under the curve.

data: (optional) *NXdata*

position: (optional) *NX_NUMBER* (Rank: dimRank) {units=*NX_ANY*}

Position values along one or more data dimensions (to hold the values for the independent variable).

intensity: (optional) *NX_NUMBER* (Rank: dimRank) {units=*NX_ANY*}

This array holds the intensity/count values of the fitted peak at each position.

Hypertext Anchors

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

- */NXpeak/data-group*
- */NXpeak/data/intensity-field*
- */NXpeak/data/position-field*
- */NXpeak/label-field*
- */NXpeak/total_area-field*

NXDL Source:

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

NXphase

Status:

base class, extends *NXObject*

Description:

Base class to describe a (thermodynamic) phase as a component of a material.

Instances of phases can be crystalline.

Symbols:

No symbol table

Groups cited:

NXatom, *NXunit_cell*

Structure:

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

Identifier for each phase.

The value 0 is reserved for the unknown phase that represents the null-model (no sufficiently significant information available). In other words, the phase_name is n/a aka notIndexed.

The identifier_phase value should match with the integer suffix of the group name which represents that instance in a NeXus/HDF5 file, i.e. if three phases were used e.g. 0, 1, and 2, three instances of *NXphase* named phase0, phase1, and phase2 should be stored in that HDF5 file.

name: (optional) *NX_CHAR*

Given name as an alias for identifying this phase.

If the identifier_phase is 0 and one would like to use the field name, the value should be n/a or notIndexed.

UNIT_CELL: (optional) *NXunit_cell*

ATOM: (optional) *NXatom*

Hypertext Anchors

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

- */NXphase/ATOM-group*
- */NXphase/name-field*
- */NXphase/phase_id-field*
- */NXphase/UNIT_CELL-group*

NXDL Source:

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

NXpid_controller

Status:

base class, extends [NXcomponent](#)

Description:

A description of a feedback system in terms of the settings of a proportional-integral-derivative (PID) controller.

Automated control of a physical quantity is often achieved by connecting the output of a sensor to an actuator (e.g. using a thermocouple to monitor the effect of a heater and influence the power provided to it). The physical quantity being operated on is typically referred to as the “Process Variable”, with the desired value being the “Setpoint” (which may vary as a function of time) and the “Error Value” is the time-varying function of the difference between the Setpoint value and the concurrent measurement of the Process Variable (Error Value = Setpoint - Process Variable).

A PID controller calculates an output value for use as an input signal to an actuator via the weighted sum of four terms:
 * Proportional: the current Error Value
 * Integral: the integral of the Error Value function
 * Derivative: the first derivative of the Error Value function
 * Feed Forward: A model of the physical system (optional)

The weightings of these terms are given by the corresponding constants: * K_p * K_i * K_d * K_ff

A classic PID controller only implements the P, I and D terms and the values of the K_p, K_i and K_d constants are sufficient to fully describe the behaviour of the feedback system implemented by such a PID controller. The inclusion of a Feed Forward term in a feedback system is a modern adaptation that aids optimisation of the automated control. It is not present in all PID controllers, but it is also not uncommon.

Note that the NXpid_controller is designed to be a child object of the actuator that its output is connected to. The parent object representing the actuator is likely to be represented by an NXactuator or NXpositioner base class, but there is a wide variety of possible applications for PID controllers.

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: (optional) [NX_NUMBER](#)

Proportional gain constant. This constant determines how strongly the output value directly follows the current Error Value. When this constant dominates, the output value is linearly proportional to the Error Value.

K_i: (optional) [NX_NUMBER](#)

Integral gain constant. This constant determines how strongly the output value should react to an accumulated offset in the Error Value that should have been corrected previously. since the integral term is proportional to both the magnitude and persistence of the Error Value over time.

K_d: (optional) *NX_NUMBER*

Derivative gain constant. This constant determines how much the feedback system should anticipate the future value of the Error Value function through adjustment of the output value that is proportional to the rate of change (i.e. derivative) of the Error Value. This term is important for damping oscillations in the feedback system.

K_ff: (optional) *NX_NUMBER*

Feed Forward gain constant. This constant determines how much the feedback system should rely on a calculated output value to achieve the desired Process Variable value. A Feed Forward system uses a model of the physical system to calculate an appropriate output value to achieve a desired Setpoint value. A description of this model should be provided in the `feed_forward_model` field.

feed_forward_model: (optional) *NX_CHAR*

A description of the model used for the Feed Forward part of the feedback system. Note that such models typically involve the Setpoint value, but not the Error Value. The simplest model is simply proportional to the Setpoint value. For example, the position (Process Variable) of a sample is measured by a linear optical encoder (sensor) and manipulated by a piezoelectric scanning stage (actuator). The corresponding Feed Forward model could be that the output value (voltage applied to the piezo) is proportional to the Setpoint value (measured position of the sample).

A complex model could involve any number of input variables, mathematical functions, and coefficients in order to describe the physical system relevant to the PID controller.

control_action: (optional) *NX_CHAR*

The Error Value of PID feedback system is normally constructed in terms of the correction needed to bring the Process Variable towards a match with the Setpoint. This “direct” control action means that a measurement of the Process Variable that is lower than the Setpoint results in a positive Error Value and a generally positive control output that tells the actuator to push the value of the Process Variable upwards. In some implementations, the actuator will respond to a more positive control output by pushing the Process Variable towards lower values (e.g. a Peltier cooler) and so the output of the feedback system must be reversed to match the behaviour of the physical system. A feedback system may also be implemented with reverse action in order to ensure that failures (e.g. disconnected sensor output or actuator input) result in a safe state (e.g. a valve should be left open to release pressure).

Any of these values: `direct` | `reverse`

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 into the PID controller.

Hypertext Anchors

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

- [/NXpid_controller/control_action-field](#)
- [/NXpid_controller/description-field](#)
- [/NXpid_controller/feed_forward_model-field](#)
- [/NXpid_controller/K_d-field](#)
- [/NXpid_controller/K_ff-field](#)
- [/NXpid_controller/K_i-field](#)
- [/NXpid_controller/K_p-field](#)
- [/NXpid_controller/pv_sensor-group](#)
- [/NXpid_controller/pv_sensor/value_log-group](#)
- [/NXpid_controller/pv_sensor/value_log/value-field](#)
- [/NXpid_controller/setpoint-field](#)

NXDL Source:

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

NXpinhole

Status:

base class, extends [NXcomponent](#)

Description:

A simple pinhole.

For more complex geometries, [NXaperture](#) should be used.

Symbols:

No symbol table

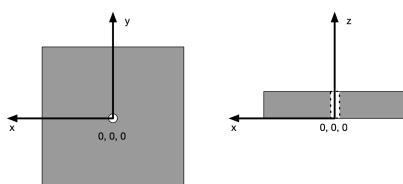
Groups cited:

none

Structure:

depends_on: (optional) [NX_CHAR](#) <=

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.

Hypertext Anchors

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

- [/NXpinhole/depends_on-field](#)
- [/NXpinhole/diameter-field](#)

NXDL Source:

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

NXpolarizer

Status:

base class, extends [NXcomponent](#)

Description:

A spin polarizer.

Symbols:

No symbol table

Groups cited:

none

Structure:

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](#) <=

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/type-field](#)

NXDL Source:

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

NXpositioner

Status:

base class, extends *NXcomponent*

Description:

A generic positioner such as a motor or piezo-electric transducer.

Symbols:

No symbol table

Groups cited:

none

Structure:

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* <=

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/value-field*](#)
- [*/NXpositioner/velocity-field*](#)

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:

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*](#)

NXDL Source:

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

NXpump

Status:

base class, extends *NXcomponent*

Description:

Device to reduce an atmosphere (real or simulated) to a controlled pressure.

Symbols:

No symbol table

Groups cited:

none

Structure:

design: (optional) *NX_CHAR*

Principle type of the pump.

Any of these values:

- membrane
- rotary_vane
- roots
- turbo_molecular

Hypertext Anchors

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

- */NXpump/design-field*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/base_classes/NXpump.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

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) <i>NX_CHAR</i>	Describes the dataset
predicted_y: (optional) <i>NX_FLOAT</i> (Rank: 1, Dimensions: [n]) {units= <i>NX_LENGTH</i> }	The y position at which the bragg peak of the reflection is predicted
@description: (optional) <i>NX_CHAR</i>	Describes the dataset
predicted_phi: (optional) <i>NX_FLOAT</i> (Rank: 1, Dimensions: [n]) {units= <i>NX_ANGLE</i> }	The phi angle at which the bragg peak of the reflection is predicted
@description: (optional) <i>NX_CHAR</i>	Describes the dataset
predicted_px_x: (optional) <i>NX_FLOAT</i> (Rank: 1, Dimensions: [n]) {units= <i>NX_UNITLESS</i> }	The x pixel position at which the bragg peak of the reflection is predicted
@description: (optional) <i>NX_CHAR</i>	Describes the dataset
predicted_px_y: (optional) <i>NX_FLOAT</i> (Rank: 1, Dimensions: [n]) {units= <i>NX_UNITLESS</i> }	The y pixel position at which the bragg peak of the reflection is predicted
@description: (optional) <i>NX_CHAR</i>	Describes the dataset
observed_frame: (optional) <i>NX_FLOAT</i> (Rank: 1, Dimensions: [n]) {units= <i>NX_UNITLESS</i> }	The estimate of the frame at which the central impact of the reflection was recorded
@description: (optional) <i>NX_CHAR</i>	Describes the dataset
observed_frame_var: (optional) <i>NX_FLOAT</i> (Rank: 1, Dimensions: [n]) {units= <i>NX_UNITLESS</i> }	The variance on the estimate of the frame at which the central impact of the reflection was recorded
@description: (optional) <i>NX_CHAR</i>	Describes the dataset
observed_frame_errors: (optional) <i>NX_FLOAT</i> (Rank: 1, Dimensions: [n]) {units= <i>NX_UNITLESS</i> }	The standard deviation of the estimate of the frame at which the central impact of the reflection was recorded
@description: (optional) <i>NX_CHAR</i>	Describes the dataset
observed_px_x: (optional) <i>NX_FLOAT</i> (Rank: 1, Dimensions: [n]) {units= <i>NX_UNITLESS</i> }	The estimate of the pixel x position at which the central impact of the reflection was recorded
@description: (optional) <i>NX_CHAR</i>	Describes the dataset
observed_px_x_var: (optional) <i>NX_FLOAT</i> (Rank: 1, Dimensions: [n]) {units= <i>NX_UNITLESS</i> }	

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@description-attribute*

NXDL Source:

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

NXregistration

Status:

base class, extends *NXprocess*

Description:

Describes image registration procedures.

Symbols:

No symbol table

Groups cited:

NXtransformations

Structure:

applied: (optional) *NX_BOOLEAN*

Has the registration been applied?

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/TRANSFORMATIONS-group*

NXDL Source:

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

NXresolution

Status:

base class, extends *NXObject*

Description:

Describes the resolution of a physical quantity.

Symbols:

No symbol table

Groups cited:

NXcalibration, *NXdata*, *NXnote*, *NXparameters*

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.

physical_quantity: (optional) *NX_CHAR*

The physical quantity of the resolution, e.g., energy, momentum, time, area, etc.

type: (optional) *NX_CHAR*

The process by which the resolution was determined.

Any of these values: `estimated` | `derived` | `calibrated` | `other`

resolution: (optional) *NX_FLOAT* {units=*NX_ANY*}

The resolution of the physical quantity.

resolution_errors: (optional) *NX_FLOAT* {units=*NX_ANY*}

Standard deviation of the resolution of the physical quantity.

relative_resolution: (optional) *NX_FLOAT* {units=*NX_ANY*}

Ratio of the resolution at a specified measurand value to that measurand value.

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

Standard deviation of the relative resolution of the physical quantity.

resolution_formula_description: (optional) *NX_CHAR*

A description of the resolution formula to determine the resolution from a set of symbols as entered by the *formula_...* fields. This should be an english description of the math used.

note: (optional) *NXnote*

Additional details of the estimate or description of the calibration procedure

response_function: (optional) *NXdata*

The response of the instrument or part of the instrument to a infinitesimally sharp input signal along the physical quantity of this group. This is also sometimes called instrument response function for time resolution or point spread function for spatial response. The resolution is typically determined by taking the full width at half maximum (FWHM) of the response function.

This could have an AXISNAME field `input` (the input axis or grid of the response function) and a DATA field `magnitude`. Both of these should have the same unit. The dimensions should match those of the *resolution* field.

formula_symbols: (optional) *NXparameters*

Symbols linking to another path in the NeXus tree to be referred to from the *resolution_formula_description* field. The TERM should be a valid path inside this application definition, i.e., of the form /entry/instrument/my_part/my_field.

CALIBRATION: (optional) *NXcalibration*

For storing details and data of a calibration to derive a resolution from data.

Hypertext Anchors

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

- */NXresolution/CALIBRATION-group*
- */NXresolution/formula_symbols-group*
- */NXresolution/note-group*
- */NXresolution/physical_quantity-field*
- */NXresolution/relative_resolution-field*
- */NXresolution/relative_resolution_errors-field*
- */NXresolution/resolution-field*
- */NXresolution/resolution_errors-field*
- */NXresolution/resolution_formula_description-field*
- */NXresolution/response_function-group*
- */NXresolution/type-field*
- */NXresolution@default-attribute*

NXDL Source:

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

NXroi

Status:

base class, extends *NXObject*

Description:

Base class to report on a region-of-interest of material analyzed or simulated.

Do not confuse this base class with *NXregion*.

Symbols:

No symbol table

Groups cited:

NXdata, NXem_ebsd, NXem_eds, NXem_eels, NXem_img, NXimage, NXprocess, NXspectrum

Structure:

PROCESS: (optional) *NXprocess*
 DATA: (optional) *NXdata*
 IMAGE: (optional) *NXimage*
 SPECTRUM: (optional) *NXspectrum*
 EM_IMG: (optional) *NXem_img*
 EM_EBSD: (optional) *NXem_ebsd*
 EM_EDS: (optional) *NXem_eds*
 EM_EELS: (optional) *NXem_eels*

Hypertext Anchors

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

- [*/NXroi/*DATA-group](#)
- [*/NXroi/*EM_EBSD-group](#)
- [*/NXroi/*EM_EDS-group](#)
- [*/NXroi/*EM_EELS-group](#)
- [*/NXroi/*EM_IMG-group](#)
- [*/NXroi/*IMAGE-group](#)
- [*/NXroi/*PROCESS-group](#)
- [*/NXroi/*SPECTRUM-group](#)

NXDL Source:

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

NXroot

Status:

base class, extends none

Description:

The root of a NeXus file.

In the NeXus standard, only NXentry groups are allowed at the root level of a file, although it is permitted to include additional groups and fields that are not part of the NeXus standard and will not be validated by NeXus tools. NeXus defines a number of root-level attributes that can be used to annotate the NeXus tree.

Note that NXroot is the only base class that does not inherit from the NXobject class, since the latter permits the inclusion of NeXus objects that are not allowed at the root level.

Symbols:

No symbol table

Groups cited:

NXentry

Structure:

@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.

Note that this is different from the version of the base class or application definition version number.

@NeXus_repository: (optional) *NX_CHAR*

A repository containing the application definitions used for creating this file. If the `NeXus_release` attribute contains a commit distance and hash, this should refer to this repository.

@NeXus_release: (optional) *NX_CHAR*

The version of NeXus definitions used in writing the file. This can either be a date-based NeXus release (e.g., YYYY.MM), see <https://github.com/nexusformat/definitions/releases> or a version tag that includes additional development information, such as a commit distance and a Git hash. This is typically formatted as `vYYYY.MM.post1.dev<commit-distance>-g<git-hash>`, where `YYYY.MM` refers to the base version of the NeXus definitions. `post1.dev<commit-distance>` indicates that the definitions are based on a commit after the base version (post1), with `<commit-distance>` being the number of commits since that version. `g<git-hash>` is the abbreviated Git hash that identifies the specific commit of the definitions being used.

If the version includes both a commit distance and a Git hash, the `NeXus_repository` attribute must be included, specifying the URL of the repository containing that version.

@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_release-attribute*
- */NXroot@NeXus_repository-attribute*
- */NXroot@NeXus_version-attribute*
- */NXroot@XML_version-attribute*

NXDL Source:

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

NXrotations

Status:

base class, extends *NXObject*

Description:

Base class to detail a set of rotations, orientations, and disorientations.

For getting a more detailed insight into the discussion of the parameterized description of orientations in materials science see:

- H.-J. Bunge
- T. B. Britton et al.
- D. Rowenhorst et al.
- A. Morawiec

Once orientations are defined, one can continue to characterize the misorientation and specifically the disorientation. The misorientation describes the rotation that is required to register the lattices of two oriented objects (like crystal lattice) into a crystallographic equivalent orientation:

- R. Bonnet

The concepts of mis- and disorientation are relevant when analyzing the crystallography of interfaces.

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 value tuples.

n_phases: How many phases with usually different crystal and symmetry are distinguished.

Groups cited:

none

Structure:

reference_frame: (optional) *NX_CHAR*

Reference to an instance of *NXcoordinate_system* which contextualizes how the here reported parameterized quantities can be interpreted.

crystal_symmetry: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_phases])

Point group which defines the symmetry of the crystal.

This has to be at least a single string. If crystal_symmetry is not provided, point group 1 is assumed.

In the case that misorientation or disorientation fields are used and the two crystal sets resolve for phases with a different crystal symmetry, this field needs to encode two strings: The first string is for phase A. The second string is for phase B. An example of this most complex case is the description of the disorientation between crystals adjoining a hetero-phase boundary.

sample_symmetry: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_phases])

Point group which defines an assumed symmetry imprinted upon processing the material/sample which could give rise to or may justify to use a simplified description of rotations, orientations, misorientations, and disorientations via numerical procedures that are known as symmetrization.

If sample_symmetry is not provided, point group 1 is assumed.

The traditionally used symmetrization operations within the texture community in Materials Science, though, have become obsolete thanks to improvements in methods, software, and available computing power.

Therefore, users are encouraged to set the sample_symmetry to 1 (triclinic).

In practice one often faces situations where indeed these assumed symmetries are anyway not fully observed, and thus an accepting of eventual inaccuracies just for the sake of reporting a simplified symmetrized description should be avoided.

rotation_quaternion: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 4]) {units=*NX_DIMENSIONLESS*}

The set of rotations expressed in quaternion parameterization considering crystal_symmetry and sample_symmetry. Rotations which should be interpreted as antipodal are not marked as such.

rotation_euler: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_ANGLE*}

The set of rotations expressed in Euler angle parameterization considering the same applied symmetries as detailed for the field rotation_quaternion. To interpret Euler angles correctly, it

is necessary to inspect the rotation conventions behind reference_frame to resolve which of the many possible Euler-angle conventions (Bunge ZXZ, XYZ, Kocks, Tait, etc.) were used.

is_antipodal: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [c])

True for all those value tuples which have assumed antipodal symmetry. False for all others.

orientation_quaternion: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 4])
 {units=*NX_DIMENSIONLESS*}

The set of orientations expressed in quaternion parameterization and obeying symmetry for equivalent cases as detailed in crystal_symmetry and sample_symmetry. The supplementary field is_antipodal can be used to mark orientations with the antipodal property.

orientation_euler: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_ANGLE*}

The set of orientations expressed in Euler angle parameterization following the same assumptions like for orientation_quaternion. To interpret Euler angles correctly, it is necessary to inspect the rotation conventions behind reference_frame to resolve which of the many Euler-angle conventions possible (Bunge ZXZ, XYZ, Kocks, Tait, etc.) were used.

misorientation_quaternion: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 4])
 {units=*NX_DIMENSIONLESS*}

The set of misorientations expressed in quaternion parameterization obeying symmetry operations for equivalent misorientations as defined by crystal_symmetry and sample_symmetry.

The misorientation should not be confused with the disorientation, as for the latter the angular argument is expected to be the minimal obeying symmetries.

misorientation_angle: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_ANGLE*}

Misorientation angular argument (eventually signed) following the same symmetry assumptions as expressed for the field misorientation_quaternion.

misorientation_axis: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3])
 {units=*NX_DIMENSIONLESS*}

Misorientation axis (normalized) and signed following the same symmetry assumptions as expressed for the field misorientation_angle.

disorientation_quaternion: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 4])
 {units=*NX_DIMENSIONLESS*}

The set of disorientations expressed in quaternion parameterization obeying symmetry operations for equivalent disorientations as defined by crystal_symmetry and sample_symmetry.

disorientation_angle: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_ANGLE*}

Disorientations angular argument (should not be signed, see D. Rowenhorst et al.) following the same symmetry assumptions as expressed for the field disorientation_quaternion.

disorientation_axis: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3])
 {units=*NX_DIMENSIONLESS*}

Disorientations axis (normalized) following the same symmetry assumptions as expressed for the field disorientation_angle.

Hypertext Anchors

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

- */NXrotations/crystal_symmetry-field*

- */NXrotations/disorientation_angle-field*
- */NXrotations/disorientation_axis-field*
- */NXrotations/disorientation_quaternion-field*
- */NXrotations/is_antipodal-field*
- */NXrotations/misorientation_angle-field*
- */NXrotations/misorientation_axis-field*
- */NXrotations/misorientation_quaternion-field*
- */NXrotations/orientation_euler-field*
- */NXrotations/orientation_quaternion-field*
- */NXrotations/reference_frame-field*
- */NXrotations/rotation_euler-field*
- */NXrotations/rotation_quaternion-field*
- */NXrotations/sample_symmetry-field*

NXDL Source:

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

NXsample

Status:

base class, extends *NXcomponent*

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*, *NXhistory*, *NXlog*, *NXoff_geometry*, *NXpositioner*, *NXsample_component*

Structure:

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_alpha_beta_gamma: (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.

physical_form: (optional) *NX_CHAR*

Physical form of the sample material. Examples include single crystal, foil, pellet, powder, thin film, disc, foam, gas, liquid, amorphous.

geometry: (optional) *NXgeometry*

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

The position and orientation of the center of mass of the sample

BEAM: (optional) *NXbeam*

Details of beam incident on sample - used to calculate sample/beam interaction point

SAMPLE_COMPONENT: (optional) *NXsample_component*

One group per sample component This is the preferred way of recording per component information over the *n_comp* arrays

transmission: (optional) *NXdata*

As a function of Wavelength

temperature_log: (optional) *NXlog*

DEPRECATED: use *temperature*, see: <https://github.com/nexusformat/definitions/issues/816>

temperature_log.value is a link to e.g. *temperature_env.sensor1.value_log.value*

temperature_env: (optional) *NXenvironment*

Additional sample temperature environment information

magnetic_field: (optional) [NXlog](#)

`magnetic_field.value` is a link to e.g. `magnetic_field_env.sensor1.value`

magnetic_field_log: (optional) [NXlog](#)

DEPRECATED: use `magnetic_field`, see: <https://github.com/nexusformat/definitions/issues/816>

`magnetic_field_log.value` is a link to e.g. `magnetic_field_env.sensor1.value_log.value`

magnetic_field_env: (optional) [NXenvironment](#)

Additional sample magnetic environment information

external_ADC: (optional) [NXlog](#)

logged value (or logic state) read from user's setup

POSITIONER: (optional) [NXpositioner](#)

Any positioner (motor, PZT, ...) used to locate the sample

OFF_GEOMETRY: (optional) [NXoff_geometry](#)

This group describes the shape of the sample

ENVIRONMENT: (optional) [NXenvironment](#)

Any environmental or external stimuli/measurements. These can include, among others: applied pressure, surrounding gas phase and gas pressure, external electric/magnetic/mechanical fields, temperature, ...

history: (optional) [NXhistory](#)

A set of physical processes that occurred to the sample prior/during experiment.

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/description-field](#)
- [/NXsample/distance-field](#)
- [/NXsample/electric_field-field](#)
- [/NXsample/electric_field@direction-attribute](#)
- [/NXsample/ENVIRONMENT-group](#)
- [/NXsample/external_ADC-group](#)
- [/NXsample/external_DAC-field](#)
- [/NXsample/geometry-group](#)

- */NXsample/history-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/physical_form-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/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*

NXDL Source:

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

NXsample_component**Status:**

base class, extends *NXcomponent*

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, *NXhistory*

Structure:

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_alpha_beta_gamma: (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

history: (optional) [NXhistory](#)

A set of physical processes that occurred to the sample component prior/during experiment.

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/history-group](#)
- [/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](#)

NXDL Source:

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

NXscanbox_em

Status:

base class, extends [NXcomponent](#)

Description:

The scan box or scan controller is a component that is used to deflect a beam of charged particles in a controlled manner.

The scan box is instructed by (an) instance(s) of *NXprogram*, some control software, which is not necessarily the same program as the one controlling other parts of the instrument.

The scanbox directs the probe of charged particles (electrons, ions) to controlled locations according to a scan scheme and plan.

Symbols:

No symbol table

Groups cited:

NXcircuit, *NXdeflector*

Structure:**scan_schema:** (optional) *NX_CHAR*

Name of the typically tech-partner-specific term that specifies an automated protocol which details how the components of the scan_box and the instrument work together to achieve a controlled scanning of the beam (over the sample surface).

Oftentimes users do not need to or are not able to disentangle the intricate details of the spatiotemporal dynamics of their instrument. Instead, often they rely on the assumption that the instrument and its controlling programs work as expected. The field scan_schema can be used to add some constraints on how the beam was scanned over the surface.

dwell_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

Time period during which the beam remains at one position.

This concept is related to term *Dwell Time* of the EMglossary standard.

flyback_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

Time period during which the beam moves from the final position of one scan line to the starting position of the subsequent scan line.

This concept is related to term *Flyback Time* of the EMglossary standard.

DEFLECTOR: (optional) *NXdeflector*

Details about components which realize the deflection technically.

This concept should be used for all those components that implement the scanning of the beam, while components like beam blankers etc. should use rather the NXdeflector concept of the NXbeam_column base class.

CIRCUIT: (optional) *NXcircuit*

Hypertext Anchors

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

- */NXscanbox_em/CIRCUIT-group*
- */NXscanbox_em/DEFLECTOR-group*
- */NXscanbox_em/dwell_time-field*
- */NXscanbox_em/flyback_time-field*
- */NXscanbox_em/scan_schema-field*

NXDL Source:

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

NXsensor**Status:**

base class, extends *NXcomponent*

Description:

A sensor used to monitor an external condition

The condition itself is described in *NXenvironment*.

Symbols:

No symbol table

Groups cited:

NXfabrication, *NXgeometry*, *NXlog*, *NXoff_geometry*, *NXorientation*

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 or a custom value (if you use a custom value, also set @custom=True):

- temperature
- pH
- magnetic_field
- electric_field
- current
- 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](#) <=

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.

FABRICATION: (optional) [NXfabrication](#) <=

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/FABRICATION-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/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:

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*

NXDL Source:

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

NXslit

Status:

base class, extends *NXcomponent*

Description:

A simple slit.

For more complex geometries, *NXaperture* should be used.

Symbols:

No symbol table

Groups cited:

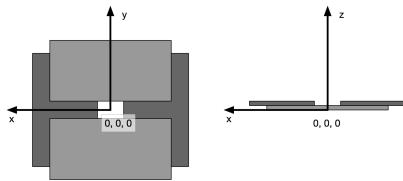
none

Structure:

depends_on: (optional) *NX_CHAR* <=

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.

Hypertext Anchors

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

- */NXslit/depends_on-field*
- */NXslit/x_gap-field*
- */NXslit/y_gap-field*

NXDL Source:

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

NXsource

Status:

base class, extends *NXcomponent*

Description:

Radiation source emitting a beam.

Examples include particle sources (electrons, neutrons, protons) or sources for electromagnetic radiation (photons). This base class can also be used to describe neutron or x-ray storage ring/facilities.

Symbols:

No symbol table

Groups cited:

NXaperture, *NXdata*, *NXdeflector*, *NXfabrication*, *NXgeometry*, *NXlens_em*, *NXnote*, *NXoff_geometry*

Structure:

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 or a custom value (if you use a custom value, also set @custom=True):

- 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
- Laser
- Dye Laser
- Broadband Tunable Light Source
- Halogen Lamp
- LED
- Mercury Cadmium Telluride Lamp
- Deuterium Lamp
- Xenon Lamp
- Globar

probe: (optional) *NX_CHAR*

type of radiation probe (pick one from the enumerated list and spell exactly)

Any of these values:

- neutron
- photon
- 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. Typically, this would be the energy of the emitted beam. For storage rings, this would be the particle beam energy.

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 or a custom value (if you use a custom value, also set @custom=True):

- 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.

wavelength: (optional) *NX_FLOAT* {units=*NX_WAVELENGTH*}

The wavelength of the radiation emitted by the source.

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

For pulsed sources, the energy of a single pulse.

peak_power: (optional) *NX_FLOAT* {units=*NX_POWER*}

For pulsed sources, the pulse energy divided by the pulse duration

anode_material: (optional) *NX_CHAR*

Material of the anode (for X-ray tubes).

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

Filament current (for X-ray tubes).

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

Emission current of the generated beam.

gas_pressure: (optional) *NX_FLOAT* {units=*NX_PRESSURE*}

Gas pressure inside ionization source.

previous_source: (optional) *NX_CHAR*

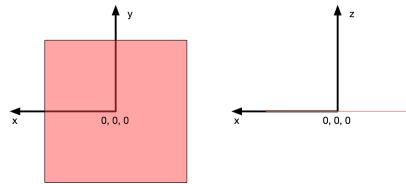
Single instance or list of instances of NXsource pointing to the sources from which a beam originated to reach this source. This can be used, for example, for secondary sources to describe which other source(s) they are derived from.

An example is the white light source in transient absorption spectroscopy, which is a supercontinuum crystal that is pumped by another laser.

In case of a primary source, this field should not be filled.

depends_on: (optional) [NX_CHAR](#) <=

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.

APERTURE: (optional) [NXaperture](#)

The size and position of an aperture inside the source.

LENS_EM: (optional) [NXlens_em](#)

Individual electromagnetic lenses inside the source.

DEFLECTOR: (optional) [NXdeflector](#)

Deflectors inside the source.

FABRICATION: (optional) [NXfabrication](#) <=

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

Hypertext Anchors

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

- [/NXsource/anode_material-field](#)
- [/NXsource/APERTURE-group](#)

- */NXsource/bunch_distance-field*
- */NXsource/bunch_length-field*
- */NXsource/bunch_pattern-group*
- */NXsource/bunch_pattern/title-field*
- */NXsource/current-field*
- */NXsource/DEFLECTOR-group*
- */NXsource/depends_on-field*
- */NXsource/distance-field*
- */NXsource/distribution-group*
- */NXsource/emission_current-field*
- */NXsource/emittance_x-field*
- */NXsource/emittance_y-field*
- */NXsource/energy-field*
- */NXsource/FABRICATION-group*
- */NXsource/filament_current-field*
- */NXsource/flux-field*
- */NXsource/frequency-field*
- */NXsource/gas_pressure-field*
- */NXsource/geometry-group*
- */NXsource/last_fill-field*
- */NXsource/last_fill@time-attribute*
- */NXsource/LENS_EM-group*
- */NXsource mode-field*
- */NXsource/name-field*
- */NXsource/name@short_name-attribute*
- */NXsource/notes-group*
- */NXsource/number_of_bunches-field*
- */NXsource/OFF_GEOMETRY-group*
- */NXsource/peak_power-field*
- */NXsource/period-field*
- */NXsource/power-field*
- */NXsource/previous_source-field*
- */NXsource/probe-field*
- */NXsource/pulse_energy-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/type-field*
- */NXsource/voltage-field*
- */NXsource/wavelength-field*

NXDL Source:

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

NXspectrum

Status:

base class, extends *NXObject*

Description:

Base class container for reporting a set of spectra.

The mostly commonly used scanning methods are supported. That is one-, two-, three-dimensional ROIs discretized using regular Euclidean tilings.

Use stack for all other tilings.

Symbols:

n_spc: Number of spectra in the stack, for stacks the slowest dimension.

n_k: Number of image points along the slower dimension (k equivalent to z).

n_j: Number of image points along the slow dimension (j equivalent to y).

n_i: Number of image points along the fast dimension (i equivalent to x).

n_energy: Number of energy bins (always the fastest dimension).

Groups cited:

NXdata, *NXnote*, *NXprocess*, *NXprogram*

Structure:

PROCESS: (optional) *NXprocess*

Details how spectra were processed from the detector readings.

mode: (optional) *NX_CHAR*

Imaging (data collection) mode of the instrument during acquisition of the data in this *NXspectrum* instance.

detector_identifier: (optional) *NX_CHAR*

Link or name of an *NXdetector* instance with which the data were collected.

input: (optional) *NXnote* <=

Resolvable data artifact (e.g. filename) from which all values in the *NXdata* instances in this *NXspectrum* were loaded during parsing.

Possibility to document from which specific other serialized resource as the source pieces of information were processed when using NeXus as a semantic file format to serialize that information differently.

The group in combination with an added field *context* therein adds context.

context: (optional) *NX_CHAR*

Reference to a location inside the artifact that points to the specific group of values that were processed if the artifacts contains several groups of values and thus further resolving of ambiguities is required.

PROGRAM: (optional) *NXprogram*

spectrum_0d: (optional) *NXdata*

One spectrum for a point of a 0d ROI. Also known as spot measurement.

intensity: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy])
 {units=*NX_UNITLESS*}

Counts

@long_name: (optional) *NX_CHAR*

Counts

axis_energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy])
 {units=*NX_ENERGY*}

Energy axis

@long_name: (optional) *NX_CHAR*

Energy

spectrum_1d: (optional) *NXdata*

One spectrum for each point of a 1d ROI.

intensity: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_i, n_energy])
 {units=*NX_UNITLESS*}

Counts

@long_name: (optional) *NX_CHAR*

Counts

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_LENGTH*}

Point coordinate along the fast dimension

@long_name: (optional) *NX_CHAR*

Point coordinate along the fast dimension

axis_energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy])
 {units=*NX_ENERGY*}

Energy axis

@long_name: (optional) *NX_CHAR*

Energy

spectrum_2d: (optional) *NXdata*

One spectrum for each scan point of 2d ROI.

intensity: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [n_j, n_i, n_energy])
{units=*NX_UNITLESS*}

Counts

@long_name: (optional) *NX_CHAR*

Counts

axis_j: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_j]) {units=*NX_LENGTH*}

Point coordinate along the slow dimension

@long_name: (optional) *NX_CHAR*

Point coordinate along the slow dimension

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_LENGTH*}

Point coordinate along the fast dimension

@long_name: (optional) *NX_CHAR*

Point coordinate along the fast dimension

axis_energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy])
{units=*NX_ENERGY*}

Energy axis

@long_name: (optional) *NX_CHAR*

Energy

spectrum_3d: (optional) *NXdata*

One spectrum for point of a 3d ROI.

intensity: (optional) *NX_NUMBER* (Rank: 4, Dimensions: [n_k, n_j, n_i, n_energy])
{units=*NX_UNITLESS*}

Counts

@long_name: (optional) *NX_CHAR*

Counts

axis_k: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_k]) {units=*NX_LENGTH*}

Point coordinate along the slower dimension

@long_name: (optional) *NX_CHAR*

Point coordinate along the slower dimension

axis_j: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_j]) {units=*NX_LENGTH*}

Point coordinate along the slow dimension

@long_name: (optional) *NX_CHAR*

Point coordinate along the slow dimension

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_LENGTH*}

Point coordinate along the fast dimension

@long_name: (optional) *NX_CHAR*

Point coordinate along the fast dimension

axis_energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy])
 {units=*NX_ENERGY*}

Energy axis

@long_name: (optional) *NX_CHAR*

Energy

stack_0d: (optional) *NXdata*

Multiple instances of spectrum_0d.

intensity: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_spc, n_energy])
 {units=*NX_UNITLESS*}

Counts

@long_name: (optional) *NX_CHAR*

Counts

indices_group: (optional) *NX_INT* (Rank: 1, Dimensions: [n_spc]) {units=*NX_UNITLESS*}

Group identifier

@long_name: (optional) *NX_CHAR*

Group identifier

indices_spectrum: (optional) *NX_INT* (Rank: 1, Dimensions: [n_spc])
 {units=*NX_UNITLESS*}

Spectrum identifier

@long_name: (optional) *NX_CHAR*

Spectrum identifier

axis_energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy])
 {units=*NX_ENERGY*}

Energy axis

@long_name: (optional) *NX_CHAR*

Energy

stack_2d: (optional) *NXdata*

Multiple instances of spectrum_2d.

intensity: (optional) *NX_NUMBER* (Rank: 4, Dimensions: [n_spc, n_j, n_i, n_energy])
 {units=*NX_UNITLESS*}

Counts

@long_name: (optional) *NX_CHAR*

Counts

indices_group: (optional) *NX_INT* (Rank: 1, Dimensions: [n_spc]) {units=*NX_UNITLESS*}

Group identifier

@long_name: (optional) *NX_CHAR*

Group identifier

indices_spectrum: (optional) *NX_INT* (Rank: 1, Dimensions: [n_spc])
{units=*NX_UNITLESS*}

Spectrum identifier

@long_name: (optional) *NX_CHAR*

Spectrum identifier

axis_j: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_j]) {units=*NX_LENGTH*}

Point coordinate along the slow dimension

@long_name: (optional) *NX_CHAR*

Point coordinate along the slow dimension

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_LENGTH*}

Point coordinate along the fast dimension

@long_name: (optional) *NX_CHAR*

Point coordinate along the fast dimension

axis_energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy])
{units=*NX_ENERGY*}

Energy axis

@long_name: (optional) *NX_CHAR*

Energy

stack_3d: (optional) *NXdata*

Multiple instances of spectrum_3d.

intensity: (optional) *NX_NUMBER* (Rank: 5, Dimensions: [n_spc, n_k, n_j, n_i, n_energy])
{units=*NX_UNITLESS*}

Counts

@long_name: (optional) *NX_CHAR*

Counts

indices_group: (optional) *NX_INT* (Rank: 1, Dimensions: [n_spc]) {units=*NX_UNITLESS*}

Group identifier

@long_name: (optional) *NX_CHAR*

Group identifier

indices_spectrum: (optional) *NX_INT* (Rank: 1, Dimensions: [n_spc])
{units=*NX_UNITLESS*}

Spectrum identifier

@long_name: (optional) *NX_CHAR*

Spectrum identifier

axis_k: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_k]) {units=*NX_LENGTH*}

Point coordinate along the slower dimension

@long_name: (optional) *NX_CHAR*

Point coordinate along the slower dimension

axis_j: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_j]) {units=*NX_LENGTH*}

Point coordinate along the slow dimension

@long_name: (optional) *NX_CHAR*

Point coordinate along the slow dimension

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_LENGTH*}

Point coordinate along the fast dimension

@long_name: (optional) *NX_CHAR*

Point coordinate along the fast dimension

axis_energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_energy]) {units=*NX_ENERGY*}

Energy axis

@long_name: (optional) *NX_CHAR*

Energy

Hypertext Anchors

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

- */NXspectrum/PROCESS-group*
- */NXspectrum/PROCESS/detector_identifier-field*
- */NXspectrum/PROCESS/input-group*
- */NXspectrum/PROCESS/input/context-field*
- */NXspectrum/PROCESS/mode-field*
- */NXspectrum/PROCESS/PROGRAM-group*
- */NXspectrum/spectrum_0d-group*
- */NXspectrum/spectrum_0d/axis_energy-field*
- */NXspectrum/spectrum_0d/axis_energy@long_name-attribute*
- */NXspectrum/spectrum_0d/intensity-field*
- */NXspectrum/spectrum_0d/intensity@long_name-attribute*
- */NXspectrum/spectrum_1d-group*
- */NXspectrum/spectrum_1d/axis_energy-field*
- */NXspectrum/spectrum_1d/axis_energy@long_name-attribute*
- */NXspectrum/spectrum_1d/axis_i-field*
- */NXspectrum/spectrum_1d/axis_i@long_name-attribute*
- */NXspectrum/spectrum_1d/intensity-field*
- */NXspectrum/spectrum_1d/intensity@long_name-attribute*
- */NXspectrum/spectrum_2d-group*
- */NXspectrum/spectrum_2d/axis_energy-field*

- */NXspectrum/spectrum_2d/axis_energy@long_name-attribute*
- */NXspectrum/spectrum_2d/axis_i-field*
- */NXspectrum/spectrum_2d/axis_i@long_name-attribute*
- */NXspectrum/spectrum_2d/axis_j-field*
- */NXspectrum/spectrum_2d/axis_j@long_name-attribute*
- */NXspectrum/spectrum_2d/intensity-field*
- */NXspectrum/spectrum_2d/intensity@long_name-attribute*
- */NXspectrum/spectrum_3d-group*
- */NXspectrum/spectrum_3d/axis_energy-field*
- */NXspectrum/spectrum_3d/axis_energy@long_name-attribute*
- */NXspectrum/spectrum_3d/axis_i-field*
- */NXspectrum/spectrum_3d/axis_i@long_name-attribute*
- */NXspectrum/spectrum_3d/axis_j-field*
- */NXspectrum/spectrum_3d/axis_j@long_name-attribute*
- */NXspectrum/spectrum_3d/axis_k-field*
- */NXspectrum/spectrum_3d/axis_k@long_name-attribute*
- */NXspectrum/spectrum_3d/intensity-field*
- */NXspectrum/spectrum_3d/intensity@long_name-attribute*
- */NXspectrum/stack_0d-group*
- */NXspectrum/stack_0d/axis_energy-field*
- */NXspectrum/stack_0d/axis_energy@long_name-attribute*
- */NXspectrum/stack_0d/indices_group-field*
- */NXspectrum/stack_0d/indices_group@long_name-attribute*
- */NXspectrum/stack_0d/indices_spectrum-field*
- */NXspectrum/stack_0d/indices_spectrum@long_name-attribute*
- */NXspectrum/stack_0d/intensity-field*
- */NXspectrum/stack_0d/intensity@long_name-attribute*
- */NXspectrum/stack_2d-group*
- */NXspectrum/stack_2d/axis_energy-field*
- */NXspectrum/stack_2d/axis_energy@long_name-attribute*
- */NXspectrum/stack_2d/axis_i-field*
- */NXspectrum/stack_2d/axis_i@long_name-attribute*
- */NXspectrum/stack_2d/axis_j-field*
- */NXspectrum/stack_2d/axis_j@long_name-attribute*
- */NXspectrum/stack_2d/indices_group-field*
- */NXspectrum/stack_2d/indices_group@long_name-attribute*

- */NXspectrum/stack_2d/indices_spectrum-field*
- */NXspectrum/stack_2d/indices_spectrum@long_name-attribute*
- */NXspectrum/stack_2d/intensity-field*
- */NXspectrum/stack_2d/intensity@long_name-attribute*
- */NXspectrum/stack_3d-group*
- */NXspectrum/stack_3d/axis_energy-field*
- */NXspectrum/stack_3d/axis_energy@long_name-attribute*
- */NXspectrum/stack_3d/axis_i-field*
- */NXspectrum/stack_3d/axis_i@long_name-attribute*
- */NXspectrum/stack_3d/axis_j-field*
- */NXspectrum/stack_3d/axis_j@long_name-attribute*
- */NXspectrum/stack_3d/axis_k-field*
- */NXspectrum/stack_3d/axis_k@long_name-attribute*
- */NXspectrum/stack_3d/indices_group-field*
- */NXspectrum/stack_3d/indices_group@long_name-attribute*
- */NXspectrum/stack_3d/indices_spectrum-field*
- */NXspectrum/stack_3d/indices_spectrum@long_name-attribute*
- */NXspectrum/stack_3d/intensity-field*
- */NXspectrum/stack_3d/intensity@long_name-attribute*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/base_classes/NXspectrum.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 Plot-table 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/thumbnil-group*
- */NXsubentry/thumbnil@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 (or to an instance of NX_coordinate_system). Following the chain may also require following depends_on links to transformations outside, for example to a common base table. If a relative path is given, it is relative to the group enclosing the depends_on specification.

For a chain of three transformations, where T_1 depends on T_2 and that in turn depends on T_3 , the final transformation T_f is

$$T_f = T_3 T_2 T_1$$

In explicit terms, the transformations are a subset of affine transformations expressed as 4x4 matrices that act on homogeneous coordinates, $w = (x, y, z, 1)^T$.

For rotation and translation,

$$T_r = \begin{pmatrix} R & o \\ 0_3 & 1 \end{pmatrix}$$
$$T_t = \begin{pmatrix} I_3 & t + o \\ 0_3 & 1 \end{pmatrix}$$

where R is the usual 3x3 rotation matrix, o is an offset vector, 0_3 is a row of 3 zeros, I_3 is the 3x3 identity matrix and t is the translation vector.

o is given by the `offset` attribute, t is given by the `vector` attribute multiplied by the field value, and R is defined as a rotation about an axis in the direction of `vector`, of angle of the field value.

NOTE

One possible use of `NXtransformations` is to define the motors and transformations for a diffractometer (goniometer). Such use is mentioned in the `NXinstrument` base class. Use one `NXtransformations` group for each diffractometer and name the group appropriate to the device. Collecting the motors of a sample table or xyz-stage in an `NXtransformations` group is equally possible.

Following the section on the general description of axis in `NXtransformations` is a section which documents the fields commonly used within NeXus for positioning purposes and their meaning. Whenever there is a need for positioning a beam line component please use the existing names. Use as many fields as needed in order to position the component. Feel free to add more axis if required. In the description given below, only those attributes which are defined through the name are specified. Add the other attributes of the full set:

- `vector`
- `offset`
- `transformation_type`
- `depends_on`

as needed.

NOTE

`NXtransformations` follows the **active** transformation convention. This means that the transformation describes how an object is moved or rotated within the coordinate system. In other words, the transformation actively changes the position or orientation of the object itself. This is in contrast to a **passive** transformation, which changes the frame of reference or coordinate system, while the object remains fixed. In case it is needed to describe multiple coordinate systems, it is strongly suggested to use the `NXcoordinate_system` base class.

Symbols:

No symbol table

Groups cited:

none

Structure:

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 specified, 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 **translation** axes the direction should be chosen for increasing displacement. For general axes, an appropriate direction should be chosen.

By default, for **rotation** axes that do not explicitly depend on a coordinate system, the direction should be chosen for a right-handed rotation with increasing angle. Note, McStas is a right handed coordinate system.

If the **NXtransformation** depends on a coordinate system (i.e., its **depends_on** attribute (or a **depends_on** further up the transformation chain) points to an instance of *NXcoordinate_system*), the rotation convention is the same as the handedness of the coordinate system (as defined by the determinant of its base vectors):

- Rotations in left-handed coordinate systems are left-handed (i.e., they follow the left-hand grip rule). In a left-handed coordinate system, positive rotation about an axis is clockwise when looking from a point on the positive axis towards its origin (from infinity towards the origin).
- Rotations in right-handed coordinate systems are right-handed (i.e., they follow the right-hand grip rule). In a right-handed coordinate system, positive rotation about an axis is counter-clockwise when looking from a point on the positive axis towards its origin (from infinity towards the origin).

Note that by using this convention, the transformation matrices in both left- and right-handed coordinate systems are the same.

@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 of a field defining the axis on which this instance of NX-transformations depends or the string “.”. It can also point to an instance of `NX_coordinate_system` if this transformation depends on it.

If it is the string “.”, it is explicitly pointing towards the default NeXus coordinate system.

@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*

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:

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](#)

NXDL Source:

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

NXunit_cell

Status:

base class, extends [NXobject](#)

Description:

Base class to describe structural aspects of an arrangement of atoms or ions including a crystallographic unit cell.

Following recommendations of [CIF](#) and [International Tables of Crystallography](#).

Symbols:

d: Dimensionality of the lattice.

Groups cited:

[NXatom](#), [NXion](#)

Structure:

reference_frame: (optional) [NX_CHAR](#)

Path to a reference frame in which the unit cell is defined to resolve ambiguity in cases when e.g. a different reference frame than the NeXus default reference frame (McStas) was chosen.

dimensionality: (optional) [NX_POSINT](#)

Dimensionality of the structure.

Any of these values: 1 | 2 | 3

a_b_c: (optional) [NX_NUMBER](#) (Rank: 1, Dimensions: [d]) {units=[NX_LENGTH](#)}

Geometry of the unit cell quantified via parameters a, b, and c.

a: (optional) [NX_NUMBER](#) {units=[NX_LENGTH](#)}

Geometry of the unit cell quantified individually via parameter a.

b: (optional) [NX_NUMBER](#) {units=[NX_LENGTH](#)}

Geometry of the unit cell quantified individually via parameter b.

c: (optional) [NX_NUMBER](#) {units=[NX_LENGTH](#)}

Geometry of the unit cell quantified individually via parameter c.

alpha_beta_gamma: (optional) [NX_NUMBER](#) (Rank: 1, Dimensions: [d]) {units=[NX_ANGLE](#)}

Geometry of the unit cell quantified via parameters alpha, beta, and gamma.

alpha: (optional) [NX_NUMBER](#) {units=[NX_ANGLE](#)}

Geometry of the unit cell quantified individually via parameter alpha.

beta: (optional) [NX_NUMBER](#) {units=[NX_ANGLE](#)}

Geometry of the unit cell quantified individually via parameter beta.

gamma: (optional) [NX_NUMBER](#) {units=[NX_ANGLE](#)}

Geometry of the unit cell quantified individually via parameter gamma.

crystal_system: (optional) [NX_CHAR](#)

Crystal system.

For a crystal system in 2D space monoclinic is an exact synonym for oblique. For a crystal system in 2D space orthorhombic is an exact synonym for rectangular. For a crystal system in 2D space tetragonal is an exact synonym for square.

Any of these values:

- triclinic
- monoclinic
- orthorhombic
- tetragonal
- rhombohedral
- hexagonal
- cubic

laue_group: (optional) [NX_CHAR](#)

Laue group using International Table of Crystallography notation.

point_group: (optional) [NX_CHAR](#)

Point group using International Table of Crystallography notation.

space_group: (optional) *NX_CHAR*

Space group from the International Table of Crystallography notation.

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.

area: (optional) *NX_NUMBER* {units=*NX_AREA*}

Area of the unit cell if dimensionality is 2.

volume: (optional) *NX_NUMBER* {units=*NX_VOLUME*}

Volume of the unit cell if dimensionality is 3.

ATOM: (optional) *NXatom*

ION: (optional) *NXion*

Hypertext Anchors

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

- */NXunit_cell/a-field*
- */NXunit_cell/a_b_c-field*
- */NXunit_cell/alpha-field*
- */NXunit_cell/alpha_beta_gamma-field*
- */NXunit_cell/area-field*
- */NXunit_cell/ATOM-group*
- */NXunit_cell/b-field*
- */NXunit_cell/beta-field*
- */NXunit_cell/c-field*
- */NXunit_cell/crystal_system-field*
- */NXunit_cell/dimensionality-field*
- */NXunit_cell/gamma-field*
- */NXunit_cell/ION-group*
- */NXunit_cell/is_centrosymmetric-field*
- */NXunit_cell/is_chiral-field*
- */NXunit_cell/laue_group-field*
- */NXunit_cell/point_group-field*

- */NXunit_cell/reference_frame-field*
- */NXunit_cell/space_group-field*
- */NXunit_cell/volume-field*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/base_classes/NXunit_cell.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:

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](#)

NXDL Source:

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

NXvelocity_selector

Status:

base class, extends [NXcomponent](#)

Description:

A neutron velocity selector

Symbols:

No symbol table

Groups cited:

[NXgeometry](#), [NXoff_geometry](#)

Structure:

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* <=

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

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/twist-field*
- */NXvelocity_selector/type-field*
- */NXvelocity_selector/wavelength-field*
- */NXvelocity_selector/wavelength_spread-field*
- */NXvelocity_selector/width-field*

NXDL Source:

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

NXxraylens**Status:**

base class, extends *NXcomponent*

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*

Structure:

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* <=

cylinder_orientation: (optional) *NXnote*

Orientation of the cylinder axis.

OFF_GEOMETRY: (optional) *NXoff_geometry*

This group describes the shape of the beam line component

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*

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

NXem

Application definition for normalized representation of electron microscopy research.

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 an indirect 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.

NXtofupd

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 bureaucracy database

instrument: (required) *NXinstrument* <=

name: (required) *NX_CHAR* <=

description: (required) *NX_CHAR*

 Brief description of the instrument

SOURCE: (required) *NXsource* <=

type: (required) *NX_CHAR* <=

 Any of these values:

- Spallation Neutron Source
- Pulsed Reactor Neutron Source
- Reactor Neutron Source
- Synchrotron X-Ray Source
- Pulsed Muon Source
- Rotating Anode X-Ray
- Fixed Tube X-Ray

name: (required) *NX_CHAR* <=

probe: (required) *NX_CHAR* <=

 Any of these values: neutron | x-ray | electron

sample: (required) *NXsample* <=

name: (required) *NX_CHAR* <=

 Descriptive name of sample

sample_id: (required) *NX_CHAR*

 Unique database id of the sample

description: (required) *NX_CHAR* <=

type: (required) *NX_CHAR* <=

 Any of these values:

- sample
- sample+can
- calibration sample
- normalisation sample

- simulated data
- none
- sample_environment

chemical_formula: (required) *NX_CHAR* <=

Chemical formula formatted according to CIF conventions

preparation_date: (required) *NX_CHAR* {units=*NX_TIME*}

situation: (required) *NX_CHAR* <=

Description of the environment the sample is in: air, vacuum, oxidizing atmosphere, dehydrated, etc.

temperature: (required) *NX_FLOAT* {units=*NX_TEMPERATURE*} <=

magnetic_field: (required) *NX_FLOAT* {units=*NX_CURRENT*} <=

electric_field: (required) *NX_FLOAT* {units=*NX_VOLTAGE*} <=

stress_field: (required) *NX_FLOAT* {units=*NX_UNITLESS*} <=

pressure: (required) *NX_FLOAT* {units=*NX_PRESSURE*} <=

Hypertext Anchors

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

- */NXarchive/ENTRY-group*
- */NXarchive/ENTRY/collection_description-field*
- */NXarchive/ENTRY/collection_identifier-field*
- */NXarchive/ENTRY/collection_time-field*
- */NXarchive/ENTRY/definition-field*
- */NXarchive/ENTRY/duration-field*
- */NXarchive/ENTRY/end_time-field*
- */NXarchive/ENTRY/entry_identifier-field*
- */NXarchive/ENTRY/experiment_description-field*
- */NXarchive/ENTRY/experiment_identifier-field*
- */NXarchive/ENTRY/instrument-group*
- */NXarchive/ENTRY/instrument/description-field*
- */NXarchive/ENTRY/instrument/name-field*
- */NXarchive/ENTRY/instrument/SOURCE-group*
- */NXarchive/ENTRY/instrument/SOURCE/name-field*
- */NXarchive/ENTRY/instrument/SOURCE/probe-field*
- */NXarchive/ENTRY/instrument/SOURCE/type-field*
- */NXarchive/ENTRY/program-field*
- */NXarchive/ENTRY/program@version-attribute*
- */NXarchive/ENTRY/release_date-field*

- */NXarchive/ENTRY/revision-field*
- */NXarchive/ENTRY/run_cycle-field*
- */NXarchive/ENTRY/sample-group*
- */NXarchive/ENTRY/sample/chemical_formula-field*
- */NXarchive/ENTRY/sample/description-field*
- */NXarchive/ENTRY/sample/electric_field-field*
- */NXarchive/ENTRY/sample/magnetic_field-field*
- */NXarchive/ENTRY/sample/name-field*
- */NXarchive/ENTRY/sample/preparation_date-field*
- */NXarchive/ENTRY/sample/pressure-field*
- */NXarchive/ENTRY/sample/sample_id-field*
- */NXarchive/ENTRY/sample/situation-field*
- */NXarchive/ENTRY/sample/stress_field-field*
- */NXarchive/ENTRY/sample/temperature-field*
- */NXarchive/ENTRY/sample/type-field*
- */NXarchive/ENTRY/start_time-field*
- */NXarchive/ENTRY/title-field*
- */NXarchive/ENTRY/user-group*
- */NXarchive/ENTRY/user/facility_user_id-field*
- */NXarchive/ENTRY/user/name-field*
- */NXarchive/ENTRY/user/role-field*
- */NXarchive/ENTRY@index-attribute*

NXDL Source:

<https://github.com/nexusformat/definitions/blob/main/applications/NXarchive.nxdl.xml>

NXarpes**Status:**

application definition, extends *NXObject*

Description:

This is an application definition for angular resolved photo electron spectroscopy.

It has been drawn up with hemispherical electron analysers in mind.

Symbols:

No symbol table

Groups cited:

NXdata, *NXdetector*, *NXentry*, *NXinstrument*, *NXmonochromator*, *NXsample*, *NXsource*

Structure:

ENTRY: (required) *NXentry*

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*

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:

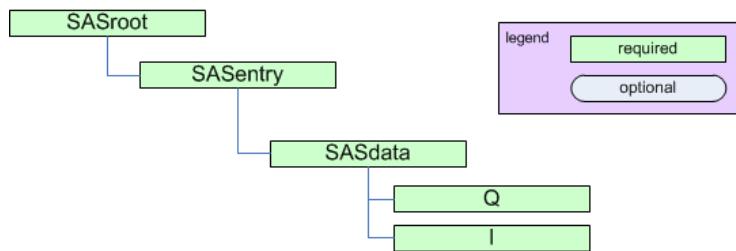


Fig. 10: The minimum requirements for *reduced* small-angle scattering data. (full image) See [below](#) for the minimum required information for a NeXus data file written to the NXcanSAS specification.

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.

```

1 NXcanSAS HDF5 data file
2   entry : NXentry
3     @NX_class = "NXentry"
4     @canSAS_class = "SASentry"
5     @version = "1.0"
6     definition = "NXcanSAS"
7     run = "<see the documentation>"
8     title = "something descriptive yet short"
9     data : NXdata
10    @NX_class = "NXdata"
11    @canSAS_class = "SASdata"
12    @signal = "I"
13    @I_axes = "<see the documentation>"
14    @Q_indices : NX_INT = <see the documentation>
15    I : NX_NUMBER
16      @units = <see the documentation>
17    Q : NX_NUMBER
18      @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 UDunits¹ 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"

Obligatory value: 1.1

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this subentry conforms.

Obligatory value: **NXcanSAS**

title: (required) *NX_CHAR* <=

Title of this *SASentry*. Make it so that you can recognize the data by its title. Could be the name of the sample, the name for the measured data, or something else representative.

run: (required) *NX_CHAR*

Run identification for this *SASentry*. For many facilities, this is an integer, such as an experiment number. Use multiple instances of *run* as needed, keeping in mind that HDF5 requires unique names for all entities in a group.

@name: (optional) *NX_CHAR*

Optional string attribute to identify this particular *run*. Could use this to associate (correlate) multiple *SASdata* elements with *run* elements.

DATA: (required) *NXdata* <=

A *SASData* group contains a single reduced small-angle scattering data set that can be represented as $I(\vec{Q})$ or $I(|\vec{Q}|)$.

Q can be either a vector (\vec{Q}) or a vector magnitude ($|\vec{Q}|$)

The name of each *SASdata* group must be unique within a *SASentry* group. Suggest using names such as **sasdata01**.

NOTE: For the first *SASdata* group, be sure to write the chosen name into the *SASentry/@default* attribute, as in:

¹ The UDunits specification also includes instructions for derived units.

`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 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 time²

Q: (required) *NX_NUMBER* {units=*NX_PER_LENGTH*}

Array of Q data to accompany I .

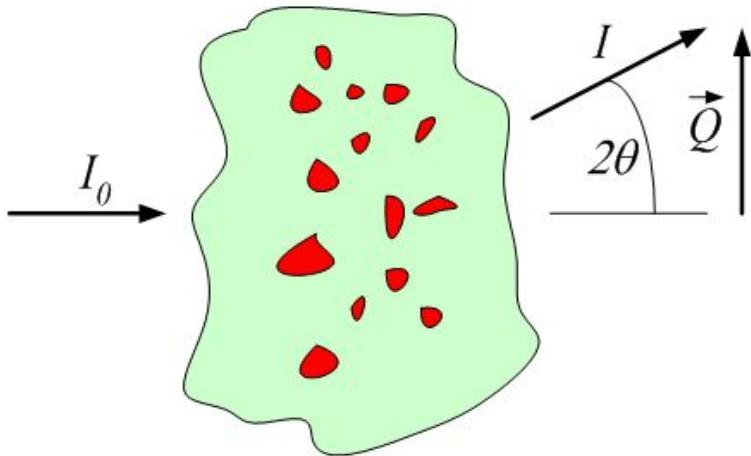


Fig. 11: The \vec{Q} geometry. (full image)

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*

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

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.

@**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 resolu-

tion 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: cm² or m²)

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 m^2/m^3 and $1/m/sr$
- $1/cm$: includes cm^2/cm^3 and $1/cm/sr$
- m^2/g
- cm^2/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 m^2/m^3 and $1/m/sr$
- `1/cm`: includes cm^2/cm^3 and $1/cm/sr$
- `m^2/g`
- `cm^2/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", "dQ1"
```

@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", "dQ1"
```

@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 sub-groups in this **NXinstrument** group. Additional information can be added but will likely be ignored by standardized data analysis 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 435, 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 435, 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: *NXdirecttoto*

INSTRUMENT: (required) *NXinstrument* <=

We definitely want the rotation_speed and energy of the chopper. Thus either a fermi_chopper or a disk_chopper group is required.

fermi_chopper: (optional) *NXfermi_chopper* <=

rotation_speed: (required) *NX_FLOAT* {units=*NX_FREQUENCY*} <=

chopper rotation speed

energy: (required) *NX_FLOAT* {units=*NX_ENERGY*} <=

energy selected

disk_chopper: (optional) *NXdisk_chopper* <=

rotation_speed: (required) *NX_FLOAT* {units=*NX_FREQUENCY*} <=

chopper rotation speed

energy: (required) *NX_FLOAT* {units=*NX_ENERGY*}

energy selected

Hypertext Anchors

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

- */NXdirecttoto/ENTRY-group*
- */NXdirecttoto/ENTRY/definition-field*
- */NXdirecttoto/ENTRY/INSTRUMENT-group*
- */NXdirecttoto/ENTRY/INSTRUMENT/disk_chopper-group*
- */NXdirecttoto/ENTRY/INSTRUMENT/disk_chopper/energy-field*
- */NXdirecttoto/ENTRY/INSTRUMENT/disk_chopper/rotation_speed-field*
- */NXdirecttoto/ENTRY/INSTRUMENT/fermi_chopper-group*
- */NXdirecttoto/ENTRY/INSTRUMENT/fermi_chopper/energy-field*
- */NXdirecttoto/ENTRY/INSTRUMENT/fermi_chopper/rotation_speed-field*
- */NXdirecttoto/ENTRY/start_time-field*
- */NXdirecttoto/ENTRY/title-field*

NXDL Source:

<https://github.com/nexusformat/definitions/blob/main/applications/NXdirecttoto.nxdl.xml>

NXem

Status:

application definition, extends *NXObject*

Description:

Application definition for normalized representation of electron microscopy research.

This application definition is a comprehensive, general description for the standardization of data and metadata collected using electron microscopy.

NXem is designed to be used for documenting experiments or computer simulations in which controlled electron beams are used to study electron-beam matter interactions, explore physical mechanisms and phenomena, or to characterize materials with an electron microscope.

The NeXus application definition NXem defines a hierarchical data model with ten building blocks:

The data model thus represents a tree of concepts with branches (groups of concepts), and either fields or attributes as leafs. NXem is defined by composing and specializing base classes into ten categories; these categories are branches:

- The field `NXem/ENTRY/definition` clarifies for NeXus files with instance data that the data schema is NXem. In combination with administrative metadata such as the `NXroot/@NeXus_version` this pinpoints which specific version of NXem the instance data in e.g. a NeXus/HDF5 file are assumed compliant with.
- The fields `NXem/ENTRY/[identifier_experiment, experiment_alias, experiment_description]` and group `NXem/ENTRY/userID` provide concepts for storing organizational metadata that contextualize the work within the research workflow.
- The fields `NXem/ENTRY/[start_time, end_time]` provide concepts for putting the research into a temporal context.
- The groups `NXem/ENTRY/[citeID, noteID]` provide concepts for adding contextual details such as citations associated with or notes, i.e. other eventually less standardized or less formalized artifacts, which nonetheless are deemed relevant when reporting about a measurement or simulation. These groups are useful when NXem is used as a serialization format for technology-partner-agnostic archival of data and metadata that have been collected during a session with an electron microscope or a simulation of such an experiment.
- The group `NXem/ENTRY/sampleID` provides concepts for storing metadata about the sample that was characterized during the session.
- The group `NXem/ENTRY/measurement` provides concepts that are useful for describing a measurement during a session with an electron microscope including the chain of events of data and metadata that were collected in such a session.
- The group ```NXem/ENTRY/simulation``` provides concepts that are useful for describing a simulation of an electron beam that interacts with matter. Combined with `NXem/ENTRY/measurement` this provides a schema for defining a digital twin of the microscope and its optical setup. For example including details about the lens setup that was assumed in the simulation or present in the microscope.
- The groups `NXem/ENTRY/[consistent_rotations, *_reference_frame*]` provide concepts for reporting coordinate systems (frames of reference) and rotation conventions that clarify how collected data (specifically those pertaining to orientable objects in the microscope, its components, or crystal and crystal defects in the material of the specimen). These metadata support interpretation for downstream data analysis pipelines and analyses which the microscope performs during a session such as indexing of diffraction patterns, image analysis in general, or compositional analyses.

- The group NXem/ENTRY/roiID provides concepts for reporting several domain- and technique-specific configurations and results of data processing that have been obtained from a measurement or a simulation.
- The group NXem/ENTRY/profiling provides concepts that are useful for reporting computational details such as programs and libraries used, the libraries of virtual environments (e.g. conda, python virtual environment), including details about the computing hardware used, and performance analyses or benchmarking of the software and its parts.

Design choices:

For the data and metadata of a microscope session to be useful, it is necessary that instance data are reported for certain concepts. These are stored typically but not only necessarily in NeXus/HDF5. Specific details how a microscope was used or its operation was simulated differ between user groups. Different peer groups and sub-domains consider different data and metadata relevant. Therefore, NXem defines constraints on the existence and cardinality of concepts and their sub-trees of concepts but seeks to offer a compromise. The key design pattern followed is that most concepts mentioned in the above-mentioned ten categories are made optional or at most recommended only. This means that it is not required to have instance data for all concepts for every use of NXem.

For most concept branches of the NXem schema we followed a design pattern where the parent is optional and its childs required or less restrictive. An example is the combination of in the optional group NXem/ENTRY/measurement having the group NXem/ENTRY/measurement/instrument is required. Thereby, users which report simulations are not forced to document the instrument but user which have characterized a sample are motivated to report about the instrument, not necessarily though about all the details of its components because also the branches below the instrument branch follow this design pattern of required-only-when-the-parent-group-is-used is picked up again in the sub branches.

Inclusive design, one schema for scanning, focused-ion beam, and transmission electron microscopes:

Contrary to many other data schemes for electron microscopy, NXem seeks to highlight the similarity of the three fundamental types of electron microscopes that are nowadays most routinely used in various research communities: An electron microscope is a beamline that provides a controlled beam of electrons and eventually beams of other particles (like ions) to investigate electron/ion matter interaction. This design of per-particle-type concept branches is realized in the base classes NXbeam_column and NXibeam_column. These provide concepts for reporting the technical components which are typically used for generating a controllable (and typically scanning) beam of particles such as electrons or ions. Thereby, focused-ion beam capabilities are modelled by adding an optional NXem/ENTRY/measurement/instrument/ibeam_column. We foresee that this design is beneficial also in the future when methods which combine photon-electron interactions via an electron microscope should be model although this proposal does not include a base class NXpbeam_column (photon-/light-beam, i.e. laser plus optical beam path descriptions and components).

We acknowledge that scanning and transmission electron microscopes are different types of instruments that have distinct differences in their electron-optical setup, and its components used. What remains the same from the perspective of an observer inside the electron-matter interaction volume within the specimen is the imaginary split into an upper and a lower half-space. In the upper a specific but eventually differently shaped electron beam illuminates the specimen. In the lower half-space the beam or particles exit the specimen or end up thermalized in thick specimens.

NXem distinguishes instrument-specific metadata based on how long individual instance data remain unchanged:

NXem/ENTRY/measurement provides two groups NXem/ENTRY/measurement/instrument and NXem/ENTRY/measurement/eventID. The first group is designed for storing metadata about the instrument which do not change over the course of the session. That is for instance the name of the technology partner who built the microscope, the microscope's serial number, the type of lenses mounted, etc. The second group is designed for metadata and data that have been collected during the microscope session. For this

specializations of `NXdata` specifically `NXimage` and `NXspectrum` are provided that can be time-stamped individually. Each instance of a group `NXem/ENTRY/measurement/eventID` contains `NXem/ENTRY/measurement/instrument` whose purpose is to store exactly those specific state settings of the microscope that was present during the collection of the event. This includes specific lens settings, apertures, etc. By virtue of design this reduces unnecessary repetition of metadata stored in the first group, which is still the frequent situation when images are reported in electron microscopy using archival formats like TIFF, PNG, etc.

NXem offers domain-specific classes for standardized reporting of method-specific configurations, data processing, and results:

These include `NXem_img` for generic and specific imaging including diffraction, `NXem_eds` for energy-dispersive X-ray spectroscopy, `NXem_ebsd` for electron backscatter diffraction, as well as `NXem_eels` for electron energy loss spectroscopy. These branches provided as examples that proof how NeXus can be used for combining session-centric data storage with data processing. These examples are naturally incomplete but show at different levels of technical depth how standardization can be useful even to report specifically formatted data representations (multi-dimensional plotting) to support that data analysis software can take advantage of standardized reporting rather than having a need for a zoo of parsers that interconvert between many representations.

Symbols:

No symbol table

Groups cited:

`NXaberration`, `NXactuator`, `NXaperture`, `NXatom`, `NXbeam`, `NXcite`, `NXcollection`, `NXcomponent`, `NXcoordinate_system`, `NXcorrector_cs`, `NXcs_profiling`, `NXdata`, `NXdeflector`, `NXdetector`, `NXbeam_column`, `NXem_ebsd`, `NXem_eds`, `NXem_eels`, `NXem_img`, `NXem_measurement`, `NXem_simulation`, `NXentry`, `NXevent_data_em`, `NXfabrication`, `NXbeam_column`, `NXimage`, `NXinstrument_em`, `NXinteraction_volume_em`, `NXion`, `NXlens_em`, `NXmanipulator`, `NXmonochromator`, `NXnote`, `NXoptical_system_em`, `NXparameters`, `NXphase`, `NXprocess`, `NXprogram`, `NXpump`, `NXroi`, `NXsample`, `NXscanbox_em`, `NXsensor`, `NXsource`, `NXspectrum`, `NXunit_cell`, `NXuser`

Structure:

`ENTRY`: (required) `NXentry`

definition: (required) `NX_CHAR` <=

Obligatory value: `NXem`

identifier_experiment: (optional) `NX_CHAR`

A (globally) unique persistent identifier for referring to this experiment.

experiment_alias: (optional) `NX_CHAR`

Alias (short name) which scientists can use to refer to this experiment.

experiment_description: (optional) `NX_CHAR` <=

Free-text description about the experiment.

Users are strongly advised to parameterize the description of their experiment by using respective groups and fields and base classes instead of writing prose into the field.

start_time: (required) `NX_DATE_TIME` <=

ISO 8601 time code with local time zone offset to UTC information included when the microscope session started. If the application demands that time codes in this section of the application definition should only be used for specifying when the experiment was performed - and the exact duration is not relevant - use this start_time field.

Often though it is useful to specify a time interval via setting both a start_time and an end_time because this enables software tools and users to collect a more detailed bookkeeping of the experiment.

Users should be aware though that even using only start_time and end_time may not be sufficient to infer how long the experiment took or for how long data were acquired. To bookkeep more fine-grained timestamps over the course of the experiment is possible with start_time and end_time fields of respective [NXevent_data_em](#) instances.

end_time: (recommended) [NX_DATE_TIME](#) <=

ISO 8601 time code with local time zone offset to UTC included when the microscope session ended.

See docstring of the start_time field to see how to use the start_time and end_time together.

profiling: (optional) [NXcs_profiling](#)

The configuration of the software that was used to generate this NeXus file.

programID: (optional) [NXprogram](#)

A collection of all programs and libraries used to generate this NeXus file. Ideally, this would enable a binary recreation from the input data.

Examples include the name and version of the libraries used to write the instance. Ideally, the software which writes these NXprogram instances also includes the version of the set of NeXus classes i.e. the specific set of base classes, application definitions, and contributed definitions with which the here described concepts can be resolved.

For the [pynxtools library](#) which is used by the [NOMAD](#) research data management system, it makes sense to store e.g. the GitHub repository commit and respective submodule references used.

Instances can also be used to document the modules and libraries that are offered by the computational environment such as those parsed from conda or python virtualenv environments.

program: (required) [NX_CHAR](#) <=

@version: (required) [NX_CHAR](#) <=

citeID: (optional) [NXcite](#)

noteID: (optional) [NXnote](#)

Collection of serialized resources associated with the experiment. Examples of such resources are files which are formatted using proprietary data models of technology partners as those generated by the control software of the microscope during the instrument session.

type: (recommended) [NX_CHAR](#) <=

file_name: (required) [NX_CHAR](#) <=

checksum: (recommended) [NX_CHAR](#) <=

algorithm: (recommended) [NX_CHAR](#) <=

userID: (optional) [NXuser](#) <=

Information about persons who performed or were involved in the microscope session or simulation run.

identifierNAME: (recommended) *NX_CHAR*

@type: (required) *NX_CHAR*

name: (optional) *NX_CHAR* <=

Given (first) name and surname.

affiliation: (optional) *NX_CHAR* <=

Name of the affiliation at the point in time when the experiment was performed.

address: (optional) *NX_CHAR* <=

Postal address of the affiliation.

email: (optional) *NX_CHAR* <=

Email address at the point in time when the experiment was performed.

Writing the most permanently used email is recommended.

telephone_number: (optional) *NX_CHAR* <=

Telephone number at the point in time when the experiment was performed.

role: (optional) *NX_CHAR* <=

User role at the point in time when the experiment was performed.

Examples are technician operating the microscope, student, postdoc, principle investigator, or guest.

sampleID: (required) *NXsample* <=

A physical entity which contains material intended to be investigated. Sample and specimen are treated as de facto synonyms. Samples can be real or virtual ones as annotated via `is_simulation`.

The suggested best practice is to call this group sample. In those cases when multiple samples need to be grouped inside one entry, these SAMPLE groups should be named using the prefix `sample` followed by an index starting from 1, i.e. (`sample1`, `sample2`).

There are at least two strategies how to store (meta)data when one analyzes multiple samples - not different ROIs on a single sample though - in one session.

One strategy is to store each sample and its results under an own NXem/ENTRY. This is one of the most frequent use cases as during most sessions typically only a single sample is investigated. In this case the name of this group should be `NXem/ENTRY/sample`.

If multiple samples are investigated storing each of them in their own ENTRY group eventually will demand unnecessary duplication of instrument details.

This can be avoided by using another strategy for storing samples and their results. Namely, by using only one instance of NXem/ENTRY. That NXem/ENTRY should then be named, like in the previous case, `NXem/entry1` and the samples should be named `sample1`, `sample2`, etc., i.e. instances should use `sample` as a name prefix.

In this case the collection of events should use `identifier_sample` to state clearly for which of the samples loaded the (characterization) event was detected.

This concept is related to term `Specimen` of the EMglossary standard.

is_simulation: (required) *NX_BOOLEAN*

Qualifier whether the sample is a real (in which case is_simulation should be set to false) or a virtual one (in which case is_simulation should be set to true).

physical_form: (recommended) [NX_CHAR](#) <=

Any of these values or a custom value (if you use a custom value, also set @custom=True): bulk | foil | thin_film | powder

identifier_sample: (recommended) [NX_CHAR](#)

Ideally, (globally) unique persistent identifier which distinguishes this sample from all others and especially the predecessor/origin from where that sample was cut off. An example of cutting off is a steel sheet that is the parent sample from which a small portion was wire-eroded that represents the sample that was then prepared for characterization with an electron microscope.

The terms sample and specimen are here considered as exact synonyms.

This field must not be used for an alias for the sample name. Instead, use name.

In cases where multiple specimens were loaded into the microscope, the identifier has to resolve the specific sample, whose results are stored by this [NXentry](#) instance, because a single NXentry should be used for the characterization of a single specimen.

Details about the specimen preparation should be stored in resources referring to identifier_parent.

@type: (required) [NX_CHAR](#)

identifier_parent: (recommended) [NX_CHAR](#)

Identifier of the sample from which the sample was cut off or the string *None*. I.e. the parent to this sample.

The purpose of this field is to support functionalities for tracking sample provenance in a research data management system.

@type: (required) [NX_CHAR](#)

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 timestamp when the measured specimen surface was actively prepared. Ideally, this matches the last timestamp that is mentioned in the digital resource pointed to by identifier_parent.

Knowing when the specimen was exposed to e.g. specific atmosphere is especially required for material that is sensitive to the environment such as specimens that were charged with fast diffusing elements or short-lived radioactive tracers.

Additional time stamps prior to preparation_date are better placed in resources which describe but do not pollute the description here with prose. Resolving these connected metadata is considered the responsibility of the research data management system and not the a NeXus file.

name: (recommended) [NX_CHAR](#) <=

Specimen name

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 research data management systems an opportunity to parse the relevant elements without having to interpret these from the resources pointed to by identifier_parent or walk through eventually deeply nested groups in individual data instances.

thickness: (optional) *NX_NUMBER* {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 typically 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_NUMBER* {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 a region-of-interest connected to individual *NXevent_data_em* instances can provide a cleaner description of the relevant details.

description: (optional) *NX_CHAR* <=

Discouraged free-text field to provide further detail.

consistent_rotations: (recommended) *NXparameters* <=

The conventions used when reporting crystal orientations. We follow the best practices of the Material Science community that are defined in reference <https://doi.org/10.1088/0965-0393/23/8/083501>.

rotation_handedness: (required) *NX_CHAR* <=

Convention how a positive rotation angle is defined when viewing from the end of the rotation unit vector towards its origin. This is in accordance with convention 2 of reference <https://doi.org/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: counter_clockwise | clockwise

rotation_convention: (required) *NX_CHAR* <=

How are rotations interpreted into an orientation according to convention 3 of reference <https://doi.org/10.1088/0965-0393/23/8/083501>.

Any of these values: passive | active

euler_angle_convention: (required) *NX_CHAR* <=

How are Euler angles interpreted given that there are several choices (e.g. zxz, xyz) according to convention 4 of reference <https://doi.org/10.1088/0965-0393/23/8/083501>.

The most frequently used convention in Materials Science is zxz, which is based on the work of H.-J. Bunge but using other conventions is possible. Proper Euler angles are distinguished from (improper) Tait-Bryan angles.

Any of these values:

- zxz
- xyx
- yzy
- zyz
- zxz
- yxy
- xyz
- yzx
- zxy
- xzy
- zyx
- yxz

axis_angle_convention: (required) *NX_CHAR* <=

To which angular range is the rotation angle argument of an axis-angle pair parameterization constrained according to convention 5 of reference <https://doi.org/10.1088/0965-0393/23/8/083501>.

Obligatory value: `rotation_angle_on_interval_zero_to_pi`

sign_convention: (required) *NX_CHAR* <=

Which sign convention is followed when converting orientations between different parametrizations/representations according to convention 6 of reference <https://doi.org/10.1088/0965-0393/23/8/083501>.

Any of these values: `p_plus_one` | `p_minus_one`

VARIADIC_reference_frame: (optional) *NXcoordinate_system*

alias: (optional) *NX_CHAR*

type: (required) *NX_CHAR* <=

origin: (recommended) *NX_CHAR* <=

x: (required) *NX_NUMBER* <=

x_direction: (recommended) *NX_CHAR* <=

y: (required) *NX_NUMBER* <=

y_direction: (recommended) *NX_CHAR* <=

z: (required) *NX_NUMBER* <=

z_direction: (recommended) *NX_CHAR* <=

processing_reference_frame: (recommended) *NXcoordinate_system*

alias: (optional) *NX_CHAR*

type: (required) *NX_CHAR* <=

origin: (recommended) *NX_CHAR* <=

Location of the origin of the processing_reference_frame.

It is assumed that 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.

If any of these assumptions is not met, the user is required to explicitly state this.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- front_top_left
- front_top_right
- front_bottom_right
- front_bottom_left
- back_top_left
- back_top_right
- back_bottom_right
- back_bottom_left

x: (required) *NX_NUMBER* <=

x_direction: (recommended) *NX_CHAR* <=

Direction of the positively pointing x-axis base vector of the processing_reference_frame.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- north
- east
- south
- west
- in
- out

y: (required) *NX_NUMBER* <=

y_direction: (recommended) *NX_CHAR* <=

Direction of the positively pointing y-axis base vector of the processing_reference_frame.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- north
- east
- south
- west
- in
- out

z: (required) *NX_NUMBER* <=

z_direction: (recommended) *NX_CHAR* <=

Direction of the positively pointing z-axis base vector of the processing_reference_frame.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- north
- east
- south
- west
- in
- out

sample_reference_frame: (recommended) *NXcoordinate_system*

depends_on: (optional) *NX_CHAR* <=

Reference to the specifically named *NXsample* instance(s) for which these conventions apply (e.g. /entry1/sample1).

alias: (optional) *NX_CHAR*

type: (required) *NX_CHAR* <=

origin: (recommended) *NX_CHAR* <=

Location of the origin of the sample_reference_frame.

It is assumed that 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.

If any of these assumptions is not met, the user is required to explicitly state this.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- front_top_left
- front_top_right
- front_bottom_right
- front_bottom_left
- back_top_left

- back_top_right
- back_bottom_right
- back_bottom_left

x: (required) *NX_NUMBER* <=

x_direction: (recommended) *NX_CHAR* <=

Direction of the positively pointing x-axis base vector of the sample_reference_frame.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- north
- east
- south
- west
- in
- out

y: (required) *NX_NUMBER* <=

y_direction: (recommended) *NX_CHAR* <=

Direction of the positively pointing y-axis base vector of the sample_reference_frame.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- north
- east
- south
- west
- in
- out

z: (required) *NX_NUMBER* <=

z_direction: (recommended) *NX_CHAR* <=

Direction of the positively pointing z-axis base vector of the sample_reference_frame.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- north
- east
- south
- west
- in

- `out`

detector_reference_frameID: (optional) *NXcoordinate_system*

The reference frame that is defined by a specific detector.

depends_on: (optional) *NX_CHAR* <=

Reference to the specifically named *NXdetector* instance for which these conventions apply (e.g. /entry1/instrument/detector1).

alias: (optional) *NX_CHAR*

type: (required) *NX_CHAR* <=

origin: (recommended) *NX_CHAR* <=

Location of the origin of the detector_reference_frame.

If the regions-of-interest forms a rectangle or cuboid, it is assumed that 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.

If any of these assumptions is not met, the user is required to explicitly state this.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `front_top_left`
- `front_top_right`
- `front_bottom_right`
- `front_bottom_left`
- `back_top_left`
- `back_top_right`
- `back_bottom_right`
- `back_bottom_left`

x: (required) *NX_NUMBER* <=

x_direction: (recommended) *NX_CHAR* <=

Direction of the positively pointing x-axis base vector of the detector_reference_frame.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `north`
- `east`
- `south`
- `west`
- `in`
- `out`

y: (required) *NX_NUMBER* <=

y_direction: (recommended) *NX_CHAR* <=

Direction of the positively pointing y-axis base vector of the detector_reference_frame.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- north
- east
- south
- west
- in
- out

z: (required) *NX_NUMBER* <=

z_direction: (recommended) *NX_CHAR* <=

Direction of the positively pointing z-axis base vector of the detector_reference_frame.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- north
- east
- south
- west
- in
- out

measurement: (optional) *NXem_measurement*

instrument: (required) *NXinstrument_em* <=

name: (recommended) *NX_CHAR* <=

location: (recommended) *NX_CHAR* <=

type: (recommended) *NX_CHAR* <=

fabrication: (required) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

programID: (recommended) *NXprogram*

Details about the control program used for operating the microscope.

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

ebeam_column: (required) *NXebeam_column* <=
fabrication: (optional) *NXfabrication* <=
vendor: (required) *NX_CHAR* <=
model: (required) *NX_CHAR* <=
serial_number: (recommended) *NX_CHAR* <=
electron_source: (recommended) *NXsource* <=
emitter_type: (required) *NX_CHAR* <=
probe: (optional) *NX_CHAR* <=
fabrication: (optional) *NXfabrication* <=
vendor: (required) *NX_CHAR* <=
model: (required) *NX_CHAR* <=
serial_number: (recommended) *NX_CHAR* <=
lensID: (optional) *NXlens_em* <=
name: (required) *NX_CHAR* <=
fabrication: (optional) *NXfabrication* <=
vendor: (required) *NX_CHAR* <=
model: (required) *NX_CHAR* <=
serial_number: (recommended) *NX_CHAR* <=
apertureID: (optional) *NXaperture* <=
name: (required) *NX_CHAR* <=
fabrication: (optional) *NXfabrication* <=
vendor: (required) *NX_CHAR* <=
model: (required) *NX_CHAR* <=
serial_number: (recommended) *NX_CHAR* <=
deflectorID: (optional) *NXdeflector* <=
name: (required) *NX_CHAR* <=
fabrication: (optional) *NXfabrication* <=
vendor: (required) *NX_CHAR* <=
model: (required) *NX_CHAR* <=
serial_number: (required) *NX_CHAR* <=
blankerID: (optional) *NXdeflector* <=
name: (required) *NX_CHAR* <=
fabrication: (optional) *NXfabrication* <=
vendor: (required) *NX_CHAR* <=
model: (required) *NX_CHAR* <=
serial_number: (required) *NX_CHAR* <=

monochromatorID: (optional) *NXmonochromator* <=

type: (required) *NX_CHAR* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

corrector_csID: (optional) *NXcorrector_cs* <=

A spherical aberration corrector is a typical component in a transmission electron microscope. Many instruments have only one, in this case the variadic suffix should be dropped. If there are multiple instances these should be numbered starting from 1, i.e. corrector_cs1, corrector_cs2.

name: (recommended) *NX_CHAR* <=

Use specifically when there are multiple instances.

fabrication: (recommended) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

corrector_ax: (optional) *NXcomponent* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

biprismID: (optional) *NXcomponent* <=

fabrication: (recommended) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

phaseplateID: (optional) *NXcomponent* <=

type: (required) *NX_CHAR* <=

fabrication: (recommended) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

sensorID: (optional) *NXsensor* <=

actuatorID: (optional) *NXactuator* <=

beamID: (optional) *NXbeam* <=

```

scan_controller: (optional) NXscanbox_em <=
fabrication: (recommended) NXfabrication <=
  vendor: (required) NX_CHAR <=
  model: (required) NX_CHAR <=
  serial_number: (recommended) NX_CHAR <=
ibeam_column: (optional) NXbeam_column <=
  fabrication: (optional) NXfabrication <=
    vendor: (required) NX_CHAR <=
    model: (required) NX_CHAR <=
    serial_number: (recommended) NX_CHAR <=
ion_source: (required) NXsource <=
  emitter_type: (required) NX_CHAR <=
  probe: (required) NXion <=
  fabrication: (optional) NXfabrication <=
    vendor: (required) NX_CHAR <=
    model: (required) NX_CHAR <=
    serial_number: (recommended) NX_CHAR <=
lensID: (optional) NXlens_em <=
  name: (required) NX_CHAR <=
  fabrication: (optional) NXfabrication <=
    vendor: (required) NX_CHAR <=
    model: (required) NX_CHAR <=
    serial_number: (recommended) NX_CHAR <=
apertureID: (optional) NXaperture <=
  name: (required) NX_CHAR <=
  fabrication: (optional) NXfabrication <=
    vendor: (required) NX_CHAR <=
    model: (required) NX_CHAR <=
    serial_number: (recommended) NX_CHAR <=
deflectorID: (optional) NXdeflector <=
  name: (required) NX_CHAR <=
  fabrication: (optional) NXfabrication <=
    vendor: (required) NX_CHAR <=
    model: (required) NX_CHAR <=
    serial_number: (recommended) NX_CHAR <=
blankerID: (optional) NXdeflector <=

```

name: (required) *NX_CHAR* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

monochromatorID: (optional) *NXmonochromator* <=

type: (required) *NX_CHAR*

name: (required) *NX_CHAR* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

sensorID: (optional) *NXsensor* <=

actuatorID: (optional) *NXactuator* <=

beamID: (optional) *NXbeam* <=

scan_controller: (optional) *NXscanbox_em* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

detectorID: (optional) *NXdetector* <=

name: (required) *NX_CHAR* <=

fabrication: (recommended) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (required) *NX_CHAR* <=

gas_injector: (optional) *NXcomponent* <=

fabrication: (recommended) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

stageID: (optional) *NXmanipulator* <=

design: (recommended) *NX_CHAR* <=

fabrication: (recommended) *NXfabrication* <=

```

vendor: (required) NX_CHAR <=
model: (required) NX_CHAR <=
serial_number: (recommended) NX_CHAR <=
nanoprobeID: (optional) NXmanipulator <=
fabrication: (recommended) NXfabrication <=
vendor: (required) NX_CHAR <=
model: (required) NX_CHAR <=
serial_number: (recommended) NX_CHAR <=
pumpID: (optional) NXpump <=
design: (required) NX_CHAR <=
sensorID: (optional) NXsensor <=
actuatorID: (optional) NXactuator <=
eventID: (optional) NXevent_data_em <=
start_time: (recommended) NX_DATE_TIME <=
end_time: (recommended) NX_DATE_TIME <=
identifier_sample: (recommended) NX_CHAR <=
imageID: (optional) NXimage <=
PROCESS: (recommended) NXprocess <=
detector_identifier: (required) NX_CHAR <=
input: (recommended) NXnote <=
type: (required) NX_CHAR <=
file_name: (required) NX_CHAR <=
checksum: (required) NX_CHAR <=
algorithm: (required) NX_CHAR <=
context: (required) NX_CHAR <=
image_1d: (optional) NXdata <=
@signal: (required) NX_CHAR <=
@axes: (required) NX_CHAR <=
@AXISNAME_indices: (required) NX_UINT
title: (recommended) NX_CHAR <=
real: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR
imag: (optional) NX_NUMBER <=
@long_name: (required) NX_CHAR
magnitude: (optional) NX_NUMBER <=
@long_name: (required) NX_CHAR <=

```

intensity: (optional) *NX_COMPLEX*
 @long_name: (required) *NX_CHAR*
 axis_i: (required) *NX_NUMBER* <=
 @long_name: (required) *NX_CHAR* <=
 image_2d: (optional) *NXdata* <=
 @signal: (required) *NX_CHAR* <=
 @axes: (required) *NX_CHAR* <=
 @AXISNAME_indices: (required) *NX_UINT*
 title: (recommended) *NX_CHAR* <=
 real: (required) *NX_NUMBER* <=
 @long_name: (required) *NX_CHAR*
 imag: (optional) *NX_NUMBER* <=
 @long_name: (required) *NX_CHAR*
 magnitude: (optional) *NX_NUMBER* <=
 @long_name: (required) *NX_CHAR* <=
 intensity: (optional) *NX_COMPLEX*
 @long_name: (required) *NX_CHAR*
 axis_j: (required) *NX_NUMBER* <=
 @long_name: (required) *NX_CHAR* <=
 axis_i: (required) *NX_NUMBER* <=
 @long_name: (required) *NX_CHAR* <=
 image_3d: (optional) *NXdata* <=
 @signal: (required) *NX_CHAR* <=
 @axes: (required) *NX_CHAR* <=
 @AXISNAME_indices: (required) *NX_UINT*
 title: (recommended) *NX_CHAR* <=
 real: (required) *NX_NUMBER* <=
 @long_name: (required) *NX_CHAR*
 imag: (optional) *NX_NUMBER* <=
 @long_name: (required) *NX_CHAR*
 magnitude: (optional) *NX_NUMBER* <=
 @long_name: (required) *NX_CHAR* <=
 intensity: (optional) *NX_COMPLEX*
 @long_name: (required) *NX_CHAR*
 axis_k: (required) *NX_NUMBER* <=
 @long_name: (required) *NX_CHAR* <=

```
axis_j: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
axis_i: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
stack_1d: (optional) NXdata <=
@signal: (required) NX_CHAR <=
@axes: (required) NX_CHAR <=
@AXISNAME_indices: (required) NX_UINT
title: (recommended) NX_CHAR <=
real: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR
imag: (optional) NX_NUMBER <=
@long_name: (required) NX_CHAR
magnitude: (optional) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
intensity: (optional) NX_NUMBER <=
@long_name: (required) NX_CHAR
indices_group: (optional) NX_INT <=
@long_name: (required) NX_CHAR <=
indices_image: (required) NX_INT <=
@long_name: (required) NX_CHAR <=
axis_i: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
stack_2d: (optional) NXdata <=
@signal: (required) NX_CHAR <=
@axes: (required) NX_CHAR <=
@AXISNAME_indices: (required) NX_UINT
title: (recommended) NX_CHAR <=
real: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR
imag: (optional) NX_NUMBER <=
@long_name: (required) NX_CHAR
magnitude: (optional) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
intensity: (optional) NX_NUMBER <=
@long_name: (required) NX_CHAR
```

indices_group: (optional) *NX_INT* <=

@long_name: (required) *NX_CHAR* <=

indices_image: (required) *NX_INT* <=

@long_name: (required) *NX_CHAR* <=

axis_j: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_i: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

stack_3d: (optional) *NXdata* <=

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_indices: (required) *NX_UINT*

title: (recommended) *NX_CHAR* <=

real: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR*

imag: (optional) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR*

magnitude: (optional) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

intensity: (optional) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR*

indices_group: (optional) *NX_INT* <=

@long_name: (required) *NX_CHAR* <=

indices_image: (required) *NX_INT* <=

@long_name: (required) *NX_CHAR* <=

axis_k: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_j: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

axis_i: (required) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR* <=

spectrumID: (optional) *NXspectrum* <=

PROCESS: (recommended) *NXprocess* <=

detector_identifier: (required) *NX_CHAR* <=

input: (recommended) *NXnote* <=

```
  type: (required) NX_CHAR <=
  file_name: (required) NX_CHAR <=
  checksum: (required) NX_CHAR <=
  algorithm: (required) NX_CHAR <=
  context: (required) NX_CHAR <=
spectrum_0d: (optional) NXdata <=
  @signal: (required) NX_CHAR <=
  @axes: (required) NX_CHAR <=
  @AXISNAME_indices: (required) NX_UINT
  title: (recommended) NX_CHAR <=
  intensity: (required) NX_NUMBER <=
    @long_name: (required) NX_CHAR <=
  axis_energy: (required) NX_NUMBER <=
    @long_name: (required) NX_CHAR <=
spectrum_1d: (optional) NXdata <=
  @signal: (required) NX_CHAR <=
  @axes: (required) NX_CHAR <=
  @AXISNAME_indices: (required) NX_UINT
  title: (recommended) NX_CHAR <=
  intensity: (required) NX_NUMBER <=
    @long_name: (required) NX_CHAR <=
  axis_i: (required) NX_NUMBER <=
    @long_name: (required) NX_CHAR <=
  axis_energy: (required) NX_NUMBER <=
    @long_name: (required) NX_CHAR <=
spectrum_2d: (optional) NXdata <=
  @signal: (required) NX_CHAR <=
  @axes: (required) NX_CHAR <=
  @AXISNAME_indices: (required) NX_UINT
  title: (recommended) NX_CHAR <=
  intensity: (required) NX_NUMBER <=
    @long_name: (required) NX_CHAR <=
  axis_j: (required) NX_NUMBER <=
    @long_name: (required) NX_CHAR <=
  axis_i: (required) NX_NUMBER <=
    @long_name: (required) NX_CHAR <=
```

```
axis_energy: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
spectrum_3d: (optional) NXdata <=
@signal: (required) NX_CHAR <=
@axes: (required) NX_CHAR <=
@AXISNAME_indices: (required) NX_UINT
title: (recommended) NX_CHAR <=
intensity: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
axis_k: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
axis_j: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
axis_i: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
axis_energy: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
stack_0d: (optional) NXdata <=
@signal: (required) NX_CHAR <=
@axes: (required) NX_CHAR <=
@AXISNAME_indices: (required) NX_UINT
title: (recommended) NX_CHAR <=
intensity: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
indices_spectrum: (required) NX_INT <=
@long_name: (required) NX_CHAR <=
axis_energy: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
stack_1d: (optional) NXdata
@signal: (required) NX_CHAR <=
@axes: (required) NX_CHAR <=
@AXISNAME_indices: (required) NX_UINT
title: (recommended) NX_CHAR <=
intensity: (required) NX_NUMBER
@long_name: (required) NX_CHAR
indices_spectrum: (required) NX_INT
```

```
    @long_name: (required) NX_CHAR
    axis_i: (required) NX_NUMBER
        @long_name: (required) NX_CHAR
        axis_energy: (required) NX_NUMBER
            @long_name: (required) NX_CHAR
            stack_2d: (optional) NXdata <=
                @signal: (required) NX_CHAR <=
                @axes: (required) NX_CHAR <=
                @AXISNAME_indices: (required) NX_UINT
                title: (recommended) NX_CHAR <=
                intensity: (required) NX_NUMBER <=
                    @long_name: (required) NX_CHAR <=
                    indices_spectrum: (required) NX_INT <=
                        @long_name: (required) NX_CHAR <=
                        axis_j: (required) NX_NUMBER <=
                            @long_name: (required) NX_CHAR <=
                            axis_i: (required) NX_NUMBER <=
                                @long_name: (required) NX_CHAR <=
                                axis_energy: (required) NX_NUMBER <=
                                    @long_name: (required) NX_CHAR <=
                                    stack_3d: (optional) NXdata <=
                                        @signal: (required) NX_CHAR <=
                                        @axes: (required) NX_CHAR <=
                                        @AXISNAME_indices: (required) NX_UINT
                                        title: (recommended) NX_CHAR <=
                                        intensity: (required) NX_NUMBER <=
                                            @long_name: (required) NX_CHAR <=
                                            indices_spectrum: (required) NX_INT <=
                                                @long_name: (required) NX_CHAR <=
                                                axis_k: (required) NX_NUMBER <=
                                                    @long_name: (required) NX_CHAR <=
                                                    axis_j: (required) NX_NUMBER <=
                                                        @long_name: (required) NX_CHAR <=
                                                        axis_i: (required) NX_NUMBER <=
                                                            @long_name: (required) NX_CHAR <=
                                                            axis_energy: (required) NX_NUMBER <=
```

@long_name: (required) *NX_CHAR* <=

instrument: (recommended) *NXinstrument_em* <=

ebeam_column: (required) *NXebeam_column* <=

operation_mode: (recommended) *NX_CHAR* <=

electron_source: (optional) *NXsource* <=

voltage: (required) *NX_NUMBER* <=

extraction_voltage: (optional) *NX_NUMBER* <=

emission_current: (optional) *NX_NUMBER* <=

filament_current: (optional) *NX_NUMBER* <=

lensID: (optional) *NXlens_em* <=

power_setting: (required) *NX_CHAR_OR_NUMBER* <=

apertureID: (optional) *NXaperture* <=

setting: (recommended) *NX_CHAR_OR_NUMBER*

Descriptor for the aperture setting when the exact technical details are unknown or not directly controllable as the control software of the microscope does not enable or was not configured to display these values for users.

deflectorID: (optional) *NXdeflector* <=

blankerID: (optional) *NXdeflector* <=

monochromatorID: (optional) *NXmonochromator* <=

applied: (required) *NX_BOOLEAN* <=

dispersion: (recommended) *NX_NUMBER* <=

voltage: (recommended) *NX_NUMBER* <=

corrector_csID: (optional) *NXcorrector_cs* <=

applied: (recommended) *NX_BOOLEAN* <=

tableauID: (required) *NXprocess* <=

c_1: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

a_1: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

b_2: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

a_2: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_3: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

s_3: (optional) *NXaberration* <=

```
    magnitude: (required) NX_NUMBER <=
    a_3: (optional) NXaberration <=
        magnitude: (required) NX_NUMBER <=
        b_4: (optional) NXaberration <=
            magnitude: (required) NX_NUMBER <=
            d_4: (optional) NXaberration <=
                magnitude: (required) NX_NUMBER <=
                a_4: (optional) NXaberration <=
                    magnitude: (required) NX_NUMBER <=
                    c_5: (optional) NXaberration <=
                        magnitude: (required) NX_NUMBER <=
                        s_5: (optional) NXaberration <=
                            magnitude: (required) NX_NUMBER <=
                            r_5: (optional) NXaberration <=
                                magnitude: (required) NX_NUMBER <=
                                a_6: (optional) NXaberration <=
                                    magnitude: (required) NX_NUMBER <=
                                    c_1_0: (optional) NXaberration <=
                                        magnitude: (required) NX_NUMBER <=
                                        c_1_2_a: (optional) NXaberration <=
                                            magnitude: (required) NX_NUMBER <=
                                            c_1_2_b: (optional) NXaberration <=
                                                magnitude: (required) NX_NUMBER <=
                                                c_2_1_a: (optional) NXaberration <=
                                                    magnitude: (required) NX_NUMBER <=
                                                    c_2_1_b: (optional) NXaberration <=
                                                        magnitude: (required) NX_NUMBER <=
                                                        c_2_3_a: (optional) NXaberration <=
                                                            magnitude: (required) NX_NUMBER <=
                                                            c_2_3_b: (optional) NXaberration <=
                                                                magnitude: (required) NX_NUMBER <=
                                                                c_3_0: (optional) NXaberration <=
                                                                    magnitude: (required) NX_NUMBER <=
                                                                    c_3_2_a: (optional) NXaberration <=
                                                                        magnitude: (required) NX_NUMBER <=
                                                                        c_3_2_b: (optional) NXaberration <=
```

magnitude: (required) *NX_NUMBER* <=

c_3_4_a: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_3_4_b: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_4_1_a: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_4_1_b: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_4_3_a: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_4_3_b: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_4_5_a: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_4_5_b: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_5_0: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_5_2_a: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_5_2_b: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_5_4_a: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_5_4_b: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_5_6_a: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

c_5_6_b: (optional) *NXaberration* <=

magnitude: (required) *NX_NUMBER* <=

corrector_ax: (optional) *NXcomponent* <=

applied: (required) *NX_BOOLEAN* <=

value_x: (required) *NX_NUMBER* <=

value_y: (required) *NX_NUMBER* <=

biprismID: (optional) *NXcomponent* <=
phaseplateID: (optional) *NXcomponent* <=
sensorID: (optional) *NXsensor* <=
actuatorID: (optional) *NXactuator* <=
beamID: (optional) *NXbeam* <=br/>
scan_controller: (recommended) *NXscanbox_em* <=br/>
scan_schema: (required) *NX_CHAR* <=br/>
dwell_time: (required) *NX_NUMBER* <=br/>
ibeam_column: (optional) *NXibeam_column* <=br/>
operation_mode: (recommended) *NX_CHAR* <=br/>
ion_source: (required) *NXsource* <=br/>
voltage: (required) *NX_NUMBER* <=br/>
flux: (required) *NX_NUMBER* <=br/>
probe: (required) *NXatom*
lensID: (optional) *NXlens_em* <=br/>
power_setting: (required) *NX_CHAR_OR_NUMBER* <=br/>
apertureID: (optional) *NXaperture* <=br/>
setting: (required) *NX_CHAR_OR_NUMBER*
Descriptor for the aperture setting when the exact technical details are unknown or not directly controllable as the control software of the microscope does not enable or was not configured to display these values for users.
deflectorID: (optional) *NXdeflector* <=br/>
blankerID: (optional) *NXdeflector* <=br/>
monochromatorID: (optional) *NXmonochromator* <=br/>
applied: (required) *NX_BOOLEAN* <=br/>
sensorID: (optional) *NXsensor* <=br/>
actuatorID: (optional) *NXactuator* <=br/>
beamID: (optional) *NXbeam* <=br/>
scan_controller: (recommended) *NXscanbox_em* <=br/>
scan_schema: (required) *NX_CHAR* <=br/>
dwell_time: (required) *NX_NUMBER* <=br/>
optics: (recommended) *NXoptical_system_em* <=br/>
detectorID: (optional) *NXdetector* <=br/>
mode: (required) *NX_CHAR*
Operation mode of the detector as displayed by the control software.

stageID: (optional) *NXmanipulator* <=

design: (recommended) *NX_CHAR* <=

tilt1: (required) *NX_NUMBER* <=

tilt2: (required) *NX_NUMBER* <=

rotation: (required) *NX_NUMBER* <=

position: (required) *NX_NUMBER* <=

sample_heater: (optional) *NXactuator* <=

physical_quantity: (required) *NX_CHAR* <=

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

Nominal current of the heater.

heater_voltage: (optional) *NX_NUMBER*
{units=*NX_VOLTAGE*}

Nominal voltage of the heater.

heater_power: (required) *NX_NUMBER*
{units=*NX_POWER*}

nanoprobeID: (optional) *NXmanipulator* <=

gas_injector: (optional) *NXcomponent* <=

pumpID: (optional) *NXpump* <=

sensorID: (optional) *NXsensor* <=

actuatorID: (optional) *NXactuator* <=

simulation: (optional) *NXem_simulation*

Documentation for a simulation of electron beam-matter interaction.

programID: (recommended) *NXprogram* <=

The program with which the simulation was performed.

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

environment: (recommended) *NXcollection*

Programs and libraries representing the computational environment

PROGRAM: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

config: (optional) *NXparameters* <=

Configuration of the simulation

results: (optional) *NXprocess* <=

Results of the simulation

IMAGE: (optional) *NXimage*

SPECTRUM: (optional) *NXspectrum*

interaction_volumeID: (optional) *NXinteraction_volume_em*

DATA: (recommended) *NXdata* <=

PROCESS: (recommended) *NXprocess* <=

roiID: (optional) *NXroi*

This concept is related to term Region Of Interest of the EMglossary standard. **img:** (optional) *NXem_img* <=

imageID: (required) *NXimage* <=

imaging_mode: (required) *NX_CHAR* <=

ebsd: (optional) *NXem_ebsd* <=

gnomonic_reference_frame: (recommended) *NXcoordinate_system* <=

alias: (optional) *NX_CHAR*

type: (required) *NX_CHAR* <=

origin: (required) *NX_CHAR* <=

x: (required) *NX_NUMBER* <=

x_direction: (recommended) *NX_CHAR* <=

y: (required) *NX_NUMBER* <=

y_direction: (recommended) *NX_CHAR* <=

z: (required) *NX_NUMBER* <=

z_direction: (recommended) *NX_CHAR* <=

pattern_center: (recommended) *NXprocess* <=

x_boundary_convention: (required) *NX_CHAR* <=

x_normalization_direction: (required) *NX_CHAR* <=

y_boundary_convention: (required) *NX_CHAR* <=

y_normalization_direction: (required) *NX_CHAR* <=

measurement: (optional) *NXprocess* <=

depends_on: (required) *NX_CHAR* <=

source: (required) *NXnote* <=

type: (recommended) *NX_CHAR* <=

file_name: (required) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

simulation: (optional) *NXprocess* <=

depends_on: (required) *NX_CHAR* <=

source: (required) *NXnote* <=

type: (recommended) *NX_CHAR* <=

file_name: (required) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

calibration: (recommended) *NXprocess* <=

indexing: (optional) *NXprocess* <=

number_of_scan_points: (required) *NX_UINT* <=

indexing_rate: (recommended) *NX_NUMBER* <=

source: (optional) *NXnote* <=

type: (recommended) *NX_CHAR* <=

file_name: (required) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

phaseID: (optional) *NXphase* <=

name: (recommended) *NX_CHAR* <=

number_of_scan_points: (required) *NX_UINT* <=

unit_cell: (required) *NXunit_cell* <=

a: (required) *NX_NUMBER* <=

b: (required) *NX_NUMBER* <=

c: (required) *NX_NUMBER* <=

alpha: (required) *NX_NUMBER* <=

beta: (required) *NX_NUMBER* <=

gamma: (required) *NX_NUMBER* <=

space_group: (required) *NX_CHAR* <=

roi: (recommended) *NXdata* <=

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_indices: (required) *NX_UINT*

title: (recommended) *NX_CHAR* <=

descriptor: (recommended) *NX_CHAR* <=

data: (required) *NX_NUMBER* <=

axis_z: (optional) *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 <=
eds: (optional) NXem_eds <=
indexing: (required) NXprocess <=
atom_types: (required) NX_CHAR <=
summary: (optional) NXdata <=
@signal: (required) NX_CHAR <=
@axes: (required) NX_CHAR <=
@AXISNAME_indices: (required) NX_UINT
title: (recommended) NX_CHAR <=
intensity: (required) NX_NUMBER <=
axis_energy: (required) NX_CHAR
@long_name: (required) NX_CHAR
ELEMENT_SPECIFIC_MAP: (optional) NXimage <=
iupac_line_candidates: (recommended) NX_CHAR <=
energy_range: (required) NX_NUMBER <=
image_2d: (required) NXdata <=
@signal: (required) NX_CHAR <=
@axes: (required) NX_CHAR <=
@AXISNAME_indices: (required) NX_UINT
title: (recommended) NX_CHAR <=
intensity: (required) NX_NUMBER <=
@units: (recommended) NX_CHAR
axis_i: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
@units: (required) NX_CHAR
axis_j: (required) NX_NUMBER <=
@long_name: (required) NX_CHAR <=
@units: (required) NX_CHAR
eels: (optional) NXem_eels <=

```

Hypertext Anchors

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

- */NXem/ENTRY-group*
- */NXem/ENTRY/citeID-group*

- */NXem/ENTRY/consistent_rotations-group*
- */NXem/ENTRY/consistent_rotations/axis_angle_convention-field*
- */NXem/ENTRY/consistent_rotations/euler_angle_convention-field*
- */NXem/ENTRY/consistent_rotations/rotation_convention-field*
- */NXem/ENTRY/consistent_rotations/rotation_handedness-field*
- */NXem/ENTRY/consistent_rotations/sign_convention-field*
- */NXem/ENTRY/definition-field*
- */NXem/ENTRY/detector_reference_frameID-group*
- */NXem/ENTRY/detector_reference_frameID/alias-field*
- */NXem/ENTRY/detector_reference_frameID/depends_on-field*
- */NXem/ENTRY/detector_reference_frameID/origin-field*
- */NXem/ENTRY/detector_reference_frameID/type-field*
- */NXem/ENTRY/detector_reference_frameID/x-field*
- */NXem/ENTRY/detector_reference_frameID/x_direction-field*
- */NXem/ENTRY/detector_reference_frameID/y-field*
- */NXem/ENTRY/detector_reference_frameID/y_direction-field*
- */NXem/ENTRY/detector_reference_frameID/z-field*
- */NXem/ENTRY/detector_reference_frameID/z_direction-field*
- */NXem/ENTRY/end_time-field*
- */NXem/ENTRY/experiment_alias-field*
- */NXem/ENTRY/experiment_description-field*
- */NXem/ENTRY/identifier_experiment-field*
- */NXem/ENTRY/measurement-group*
- */NXem/ENTRY/measurement/eventID-group*
- */NXem/ENTRY/measurement/eventID/end_time-field*
- */NXem/ENTRY/measurement/eventID/identifier_sample-field*
- */NXem/ENTRY/measurement/eventID/imageID-group*
- */NXem/ENTRY/measurement/eventID/imageID/image_1d-group*
- */NXem/ENTRY/measurement/eventID/imageID/image_1d/axis_i-field*
- */NXem/ENTRY/measurement/eventID/imageID/image_1d/axis_i@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/image_1d/imag-field*
- */NXem/ENTRY/measurement/eventID/imageID/image_1d/imag@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/image_1d/intensity-field*
- */NXem/ENTRY/measurement/eventID/imageID/image_1d/intensity@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/image_1d/magnitude-field*
- */NXem/ENTRY/measurement/eventID/imageID/image_1d/magnitude@long_name-attribute*

- /NXem/ENTRY/measurement/eventID/imageID/image_1d/real-field
- /NXem/ENTRY/measurement/eventID/imageID/image_1d/real@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_1d/title-field
- /NXem/ENTRY/measurement/eventID/imageID/image_1d@axes-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_1d@AXISNAME_indices-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_1d@signal-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_2d-group
- /NXem/ENTRY/measurement/eventID/imageID/image_2d/axis_i-field
- /NXem/ENTRY/measurement/eventID/imageID/image_2d/axis_i@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_2d/axis_j-field
- /NXem/ENTRY/measurement/eventID/imageID/image_2d/axis_j@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_2d/imag-field
- /NXem/ENTRY/measurement/eventID/imageID/image_2d/imag@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_2d/intensity-field
- /NXem/ENTRY/measurement/eventID/imageID/image_2d/intensity@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_2d/magnitude-field
- /NXem/ENTRY/measurement/eventID/imageID/image_2d/magnitude@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_2d/real-field
- /NXem/ENTRY/measurement/eventID/imageID/image_2d/real@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_2d/title-field
- /NXem/ENTRY/measurement/eventID/imageID/image_2d@axes-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_2d@AXISNAME_indices-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_2d@signal-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_3d-group
- /NXem/ENTRY/measurement/eventID/imageID/image_3d/axis_i-field
- /NXem/ENTRY/measurement/eventID/imageID/image_3d/axis_i@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_3d/axis_j-field
- /NXem/ENTRY/measurement/eventID/imageID/image_3d/axis_j@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_3d/axis_k-field
- /NXem/ENTRY/measurement/eventID/imageID/image_3d/axis_k@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_3d/imag-field
- /NXem/ENTRY/measurement/eventID/imageID/image_3d/imag@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_3d/intensity-field
- /NXem/ENTRY/measurement/eventID/imageID/image_3d/intensity@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_3d/magnitude-field
- /NXem/ENTRY/measurement/eventID/imageID/image_3d/magnitude@long_name-attribute

- */NXem/ENTRY/measurement/eventID/imageID/image_3d/real-field*
- */NXem/ENTRY/measurement/eventID/imageID/image_3d/real@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/image_3d/title-field*
- */NXem/ENTRY/measurement/eventID/imageID/image_3d@axes-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/image_3d@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/image_3d@signal-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/PROCESS-group*
- */NXem/ENTRY/measurement/eventID/imageID/PROCESS/detector_identifier-field*
- */NXem/ENTRY/measurement/eventID/imageID/PROCESS/input-group*
- */NXem/ENTRY/measurement/eventID/imageID/PROCESS/input/algorithm-field*
- */NXem/ENTRY/measurement/eventID/imageID/PROCESS/input/checksum-field*
- */NXem/ENTRY/measurement/eventID/imageID/PROCESS/input/context-field*
- */NXem/ENTRY/measurement/eventID/imageID/PROCESS/input/file_name-field*
- */NXem/ENTRY/measurement/eventID/imageID/PROCESS/input/type-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d-group*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/axis_i-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/axis_i@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/imag-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/imag@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/indices_group-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/indices_group@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/indices_image-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/indices_image@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/intensity-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/intensity@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/magnitude-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/magnitude@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/real-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/real@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d/title-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d@axes-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_1d@signal-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d-group*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/axis_i-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/axis_i@long_name-attribute*

- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/axis_j-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/axis_j@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/imag-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/imag@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/indices_group-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/indices_group@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/indices_image-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/indices_image@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/intensity-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/intensity@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/magnitude-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/magnitude@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/real-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/real@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d/title-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d@axes-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_2d@signal-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d-group*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/axis_i-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/axis_i@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/axis_j-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/axis_j@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/axis_k-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/axis_k@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/imag-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/imag@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/indices_group-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/indices_group@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/indices_image-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/indices_image@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/intensity-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/intensity@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/magnitude-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/magnitude@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/real-field*

- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/real@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d/title-field*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d@axes-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/eventID/imageID/stack_3d@signal-attribute*
- */NXem/ENTRY/measurement/eventID/instrument-group*
- */NXem/ENTRY/measurement/eventID/instrument/actuatorID-group*
- */NXem/ENTRY/measurement/eventID/instrument/detectorID-group*
- */NXem/ENTRY/measurement/eventID/instrument/detectorID mode-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/actuatorID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/apertureID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/apertureID/setting-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/beamID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/biprismID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/blankerID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_ax-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_ax/applied-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_ax/value_x-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_ax/value_y-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/applied-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/a_1-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/a_1/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/a_2-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/a_2/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/a_3-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/a_3/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/a_4-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/a_4/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/a_6-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/a_6/magnitude-field*

- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/b_2-group`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/b_2/magnitude-field`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/b_4-group`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/b_4/magnitude-field`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_1-group`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_1/magnitude-field`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_1_0-group`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_1_0/magnitude-field`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_1_2_a-group`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_1_2_a/magnitude-field`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_1_2_b-group`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_1_2_b/magnitude-field`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_2_1_a-group`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_2_1_a/magnitude-field`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_2_1_b-group`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_2_1_b/magnitude-field`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_2_3_a-group`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_2_3_a/magnitude-field`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_2_3_b-group`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_2_3_b/magnitude-field`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_3-group`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_3/magnitude-field`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_3_0-group`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_3_0/magnitude-field`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_3_2_a-group`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_3_2_a/magnitude-field`
- `/NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_3_2_b-group`

- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_3_2_b/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_3_4_a-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_3_4_a/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_3_4_b-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_3_4_b/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_4_1_a-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_4_1_a/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_4_1_b-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_4_1_b/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_4_3_a-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_4_3_a/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_4_3_b-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_4_3_b/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_4_5_a-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_4_5_a/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_4_5_b-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_4_5_b/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5_0-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5_0/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5_2_a-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5_2_a/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5_2_b-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5_2_b/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5_4_a-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5_4_a/magnitude-field*

- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5_4_b-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5_4_b/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5_6_a-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5_6_a/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5_6_b-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/c_5_6_b/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/d_4-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/d_4/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/r_5-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/r_5/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/s_3-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/s_3/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/s_5-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/corrector_csID/tableauID/s_5/magnitude-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/deflectorID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/electron_source-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/electron_source/emission_current-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/electron_source/extraction_voltage-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/electron_source/filament_current-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/electron_source/voltage-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/lensID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/lensID/power_setting-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/monochromatorID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/monochromatorID/applied-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/monochromatorID/dispersion-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/monochromatorID/voltage-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/operation_mode-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/phaseplateID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/scan_controller-group*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/scan_controller/dwell_time-field*
- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/scan_controller/scan_schema-field*

- */NXem/ENTRY/measurement/eventID/instrument/ebeam_column/sensorID-group*
- */NXem/ENTRY/measurement/eventID/instrument/gas_injector-group*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column-group*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/actuatorID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/apertureID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/apertureID/setting-field*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/beamID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/blankerID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/deflectorID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/ion_source-group*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/ion_source/flux-field*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/ion_source/probe-group*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/ion_source/voltage-field*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/lensID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/lensID/power_setting-field*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/monochromatorID-group*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/monochromatorID/applied-field*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/operation_mode-field*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/scan_controller-group*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/scan_controller/dwell_time-field*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/scan_controller/scan_schema-field*
- */NXem/ENTRY/measurement/eventID/instrument/ibeam_column/sensorID-group*
- */NXem/ENTRY/measurement/eventID/instrument/nanoprobeID-group*
- */NXem/ENTRY/measurement/eventID/instrument/optics-group*
- */NXem/ENTRY/measurement/eventID/instrument/pumpID-group*
- */NXem/ENTRY/measurement/eventID/instrument/sensorID-group*
- */NXem/ENTRY/measurement/eventID/instrument/stageID-group*
- */NXem/ENTRY/measurement/eventID/instrument/stageID/design-field*
- */NXem/ENTRY/measurement/eventID/instrument/stageID/position-field*
- */NXem/ENTRY/measurement/eventID/instrument/stageID/rotation-field*
- */NXem/ENTRY/measurement/eventID/instrument/stageID/sample_heater-group*
- */NXem/ENTRY/measurement/eventID/instrument/stageID/sample_heater/heater_current-field*
- */NXem/ENTRY/measurement/eventID/instrument/stageID/sample_heater/heater_power-field*
- */NXem/ENTRY/measurement/eventID/instrument/stageID/sample_heater/heater_voltage-field*
- */NXem/ENTRY/measurement/eventID/instrument/stageID/sample_heater/physical_quantity-field*
- */NXem/ENTRY/measurement/eventID/instrument/stageID/tilt1-field*

- */NXem/ENTRY/measurement/eventID/instrument/stageID/tilt2-field*
- */NXem/ENTRY/measurement/eventID/spectrumID-group*
- */NXem/ENTRY/measurement/eventID/spectrumID/PROCESS-group*
- */NXem/ENTRY/measurement/eventID/spectrumID/PROCESS/detector_identifier-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/PROCESS/input-group*
- */NXem/ENTRY/measurement/eventID/spectrumID/PROCESS/input/algorithm-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/PROCESS/input/checksum-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/PROCESS/input/context-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/PROCESS/input/file_name-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/PROCESS/input/type-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_0d-group*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_0d/axis_energy-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_0d/axis_energy@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_0d/intensity-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_0d/intensity@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_0d/title-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_0d@axes-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_0d@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_0d@signal-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_1d-group*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_1d/axis_energy-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_1d/axis_energy@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_1d/axis_i-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_1d/axis_i@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_1d/intensity-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_1d/intensity@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_1d/title-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_1d@axes-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_1d@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_1d@signal-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_2d-group*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_2d/axis_energy-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_2d/axis_energy@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_2d/axis_i-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_2d/axis_i@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_2d/axis_j-field*

- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_2d/axis_j@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_2d/intensity-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_2d/intensity@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_2d/title-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_2d@axes-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_2d@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_2d@signal-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d-group*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d/axis_energy-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d/axis_energy@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d/axis_i-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d/axis_i@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d/axis_j-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d/axis_j@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d/axis_k-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d/axis_k@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d/intensity-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d/intensity@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d/title-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d@axes-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/spectrum_3d@signal-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_0d-group*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_0d/axis_energy-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_0d/axis_energy@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_0d/indices_spectrum-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_0d/indices_spectrum@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_0d/intensity-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_0d/intensity@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_0d/title-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_0d@axes-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_0d@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_0d@signal-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_1d-group*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_1d/axis_energy-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_1d/axis_energy@long_name-attribute*

- */NXem/ENTRY/measurement/eventID/spectrumID/stack_1d/axis_i-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_1d/axis_i@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_1d/indices_spectrum-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_1d/indices_spectrum@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_1d/intensity-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_1d/intensity@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_1d/title-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_1d@axes-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_1d@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_1d@signal-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d-group*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d/axis_energy-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d/axis_energy@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d/axis_i-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d/axis_i@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d/axis_j-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d/axis_j@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d/indices_spectrum-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d/indices_spectrum@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d/intensity-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d/intensity@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d/title-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d@axes-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_2d@signal-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d-group*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d/axis_energy-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d/axis_energy@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d/axis_i-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d/axis_i@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d/axis_j-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d/axis_j@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d/axis_k-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d/axis_k@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d/indices_spectrum-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d/indices_spectrum@long_name-attribute*

- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d/intensity-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d/intensity@long_name-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d/title-field*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d@axes-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d@AXISNAME_indices-attribute*
- */NXem/ENTRY/measurement/eventID/spectrumID/stack_3d@signal-attribute*
- */NXem/ENTRY/measurement/eventID/start_time-field*
- */NXem/ENTRY/measurement/instrument-group*
- */NXem/ENTRY/measurement/instrument/actuatorID-group*
- */NXem/ENTRY/measurement/instrument/detectorID-group*
- */NXem/ENTRY/measurement/instrument/detectorID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/detectorID/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/detectorID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/detectorID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/detectorID/name-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/actuatorID-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/apertureID-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/apertureID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/apertureID/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/apertureID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/apertureID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/apertureID/name-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/beamID-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/biprismID-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/biprismID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/biprismID/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/biprismID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/biprismID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/blankerID-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/blankerID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/blankerID/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/blankerID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/blankerID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/blankerID/name-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/corrector_ax-group*

- */NXem/ENTRY/measurement/instrument/ebeam_column/corrector_ax/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/corrector_ax/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/corrector_ax/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/corrector_ax/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/corrector_csID-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/corrector_csID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/corrector_csID/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/corrector_csID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/corrector_csID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/corrector_csID/name-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/deflectorID-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/deflectorID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/deflectorID/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/deflectorID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/deflectorID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/deflectorID/name-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/electron_source-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/electron_source/emitter_type-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/electron_source/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/electron_source/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/electron_source/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/electron_source/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/electron_source/probe-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/lensID-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/lensID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/lensID/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/lensID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/lensID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/lensID/name-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/monochromatorID-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/monochromatorID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/monochromatorID/fabrication/model-field*

- */NXem/ENTRY/measurement/instrument/ebeam_column/monochromatorID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/monochromatorID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/monochromatorID/type-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/phaseplateID-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/phaseplateID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/phaseplateID/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/phaseplateID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/phaseplateID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/phaseplateID/type-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/scan_controller-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/scan_controller/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ebeam_column/scan_controller/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/scan_controller/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/scan_controller/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ebeam_column/sensorID-group*
- */NXem/ENTRY/measurement/instrument/fabrication-group*
- */NXem/ENTRY/measurement/instrument/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/gas_injector-group*
- */NXem/ENTRY/measurement/instrument/gas_injector/fabrication-group*
- */NXem/ENTRY/measurement/instrument/gas_injector/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/gas_injector/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/gas_injector/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/actuatorID-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/apertureID-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/apertureID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/apertureID/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/apertureID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/apertureID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/apertureID/name-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/beamID-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/blankerID-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/blankerID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/blankerID/fabrication/model-field*

- */NXem/ENTRY/measurement/instrument/ibeam_column/blankerID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/blankerID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/blankerID/name-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/deflectorID-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/deflectorID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/deflectorID/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/deflectorID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/deflectorID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/deflectorID/name-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/ion_source-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/ion_source/emitter_type-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/ion_source/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/ion_source/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/ion_source/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/ion_source/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/ion_source/probe-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/lensID-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/lensID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/lensID/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/lensID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/lensID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/lensID/name-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/monochromatorID-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/monochromatorID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/monochromatorID/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/monochromatorID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/monochromatorID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/monochromatorID/name-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/monochromatorID/type-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/scan_controller-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/scan_controller/fabrication-group*
- */NXem/ENTRY/measurement/instrument/ibeam_column/scan_controller/fabrication/model-field*

- */NXem/ENTRY/measurement/instrument/ibeam_column/scan_controller/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/scan_controller/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/ibeam_column/sensorID-group*
- */NXem/ENTRY/measurement/instrument/location-field*
- */NXem/ENTRY/measurement/instrument/name-field*
- */NXem/ENTRY/measurement/instrument/nanoprobeID-group*
- */NXem/ENTRY/measurement/instrument/nanoprobeID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/nanoprobeID/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/nanoprobeID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/nanoprobeID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/programID-group*
- */NXem/ENTRY/measurement/instrument/programID/program-field*
- */NXem/ENTRY/measurement/instrument/programID/program@version-attribute*
- */NXem/ENTRY/measurement/instrument/pumpID-group*
- */NXem/ENTRY/measurement/instrument/pumpID/design-field*
- */NXem/ENTRY/measurement/instrument/sensorID-group*
- */NXem/ENTRY/measurement/instrument/stageID-group*
- */NXem/ENTRY/measurement/instrument/stageID/design-field*
- */NXem/ENTRY/measurement/instrument/stageID/fabrication-group*
- */NXem/ENTRY/measurement/instrument/stageID/fabrication/model-field*
- */NXem/ENTRY/measurement/instrument/stageID/fabrication/serial_number-field*
- */NXem/ENTRY/measurement/instrument/stageID/fabrication/vendor-field*
- */NXem/ENTRY/measurement/instrument/type-field*
- */NXem/ENTRY/noteID-group*
- */NXem/ENTRY/noteID/algorithm-field*
- */NXem/ENTRY/noteID/checksum-field*
- */NXem/ENTRY/noteID/file_name-field*
- */NXem/ENTRY/noteID/type-field*
- */NXem/ENTRY/processing_reference_frame-group*
- */NXem/ENTRY/processing_reference_frame/alias-field*
- */NXem/ENTRY/processing_reference_frame/origin-field*
- */NXem/ENTRY/processing_reference_frame/type-field*
- */NXem/ENTRY/processing_reference_frame/x-field*
- */NXem/ENTRY/processing_reference_frame/x_direction-field*
- */NXem/ENTRY/processing_reference_frame/y-field*
- */NXem/ENTRY/processing_reference_frame/y_direction-field*

- /NXem/ENTRY/processing_reference_frame/z-field
- /NXem/ENTRY/processing_reference_frame/z_direction-field
- /NXem/ENTRY/profiling-group
- /NXem/ENTRY/profiling/programID-group
- /NXem/ENTRY/profiling/programID/program-field
- /NXem/ENTRY/profiling/programID/program@version-attribute
- /NXem/ENTRY/roiID-group
- /NXem/ENTRY/roiID/ebsd-group
- /NXem/ENTRY/roiID/ebsd/calibration-group
- /NXem/ENTRY/roiID/ebsd/gnomonic_reference_frame-group
- /NXem/ENTRY/roiID/ebsd/gnomonic_reference_frame/alias-field
- /NXem/ENTRY/roiID/ebsd/gnomonic_reference_frame/origin-field
- /NXem/ENTRY/roiID/ebsd/gnomonic_reference_frame/type-field
- /NXem/ENTRY/roiID/ebsd/gnomonic_reference_frame/x-field
- /NXem/ENTRY/roiID/ebsd/gnomonic_reference_frame/x_direction-field
- /NXem/ENTRY/roiID/ebsd/gnomonic_reference_frame/y-field
- /NXem/ENTRY/roiID/ebsd/gnomonic_reference_frame/y_direction-field
- /NXem/ENTRY/roiID/ebsd/gnomonic_reference_frame/z-field
- /NXem/ENTRY/roiID/ebsd/gnomonic_reference_frame/z_direction-field
- /NXem/ENTRY/roiID/ebsd/indexing-group
- /NXem/ENTRY/roiID/ebsd/indexing/indexing_rate-field
- /NXem/ENTRY/roiID/ebsd/indexing/number_of_scan_points-field
- /NXem/ENTRY/roiID/ebsd/indexing/phaseID-group
- /NXem/ENTRY/roiID/ebsd/indexing/phaseID/name-field
- /NXem/ENTRY/roiID/ebsd/indexing/phaseID/number_of_scan_points-field
- /NXem/ENTRY/roiID/ebsd/indexing/phaseID/unit_cell-group
- /NXem/ENTRY/roiID/ebsd/indexing/phaseID/unit_cell/a-field
- /NXem/ENTRY/roiID/ebsd/indexing/phaseID/unit_cell/alpha-field
- /NXem/ENTRY/roiID/ebsd/indexing/phaseID/unit_cell/b-field
- /NXem/ENTRY/roiID/ebsd/indexing/phaseID/unit_cell/beta-field
- /NXem/ENTRY/roiID/ebsd/indexing/phaseID/unit_cell/c-field
- /NXem/ENTRY/roiID/ebsd/indexing/phaseID/unit_cell/gamma-field
- /NXem/ENTRY/roiID/ebsd/indexing/phaseID/unit_cell/space_group-field
- /NXem/ENTRY/roiID/ebsd/indexing/roi-group
- /NXem/ENTRY/roiID/ebsd/indexing/roi/axis_x-field
- /NXem/ENTRY/roiID/ebsd/indexing/roi/axis_x@long_name-attribute

- */NXem/ENTRY/roiID/ebsd/indexing/roi/axis_y-field*
- */NXem/ENTRY/roiID/ebsd/indexing/roi/axis_y@long_name-attribute*
- */NXem/ENTRY/roiID/ebsd/indexing/roi/axis_z-field*
- */NXem/ENTRY/roiID/ebsd/indexing/roi/axis_z@long_name-attribute*
- */NXem/ENTRY/roiID/ebsd/indexing/roi/data-field*
- */NXem/ENTRY/roiID/ebsd/indexing/roi/descriptor-field*
- */NXem/ENTRY/roiID/ebsd/indexing/roi/title-field*
- */NXem/ENTRY/roiID/ebsd/indexing/roi@axes-attribute*
- */NXem/ENTRY/roiID/ebsd/indexing/roi@AXISNAME_indices-attribute*
- */NXem/ENTRY/roiID/ebsd/indexing/roi@signal-attribute*
- */NXem/ENTRY/roiID/ebsd/indexing/source-group*
- */NXem/ENTRY/roiID/ebsd/indexing/source/algorithm-field*
- */NXem/ENTRY/roiID/ebsd/indexing/source/checksum-field*
- */NXem/ENTRY/roiID/ebsd/indexing/source/file_name-field*
- */NXem/ENTRY/roiID/ebsd/indexing/source/type-field*
- */NXem/ENTRY/roiID/ebsd/measurement-group*
- */NXem/ENTRY/roiID/ebsd/measurement/depends_on-field*
- */NXem/ENTRY/roiID/ebsd/measurement/source-group*
- */NXem/ENTRY/roiID/ebsd/measurement/source/algorithm-field*
- */NXem/ENTRY/roiID/ebsd/measurement/source/checksum-field*
- */NXem/ENTRY/roiID/ebsd/measurement/source/file_name-field*
- */NXem/ENTRY/roiID/ebsd/measurement/source/type-field*
- */NXem/ENTRY/roiID/ebsd/pattern_center-group*
- */NXem/ENTRY/roiID/ebsd/pattern_center/x_boundary_convention-field*
- */NXem/ENTRY/roiID/ebsd/pattern_center/x_normalization_direction-field*
- */NXem/ENTRY/roiID/ebsd/pattern_center/y_boundary_convention-field*
- */NXem/ENTRY/roiID/ebsd/pattern_center/y_normalization_direction-field*
- */NXem/ENTRY/roiID/ebsd/simulation-group*
- */NXem/ENTRY/roiID/ebsd/simulation/depends_on-field*
- */NXem/ENTRY/roiID/ebsd/simulation/source-group*
- */NXem/ENTRY/roiID/ebsd/simulation/source/algorithm-field*
- */NXem/ENTRY/roiID/ebsd/simulation/source/checksum-field*
- */NXem/ENTRY/roiID/ebsd/simulation/source/file_name-field*
- */NXem/ENTRY/roiID/ebsd/simulation/source/type-field*
- */NXem/ENTRY/roiID/eds-group*
- */NXem/ENTRY/roiID/eds/indexing-group*

- */NXem/ENTRY/roiID/eds/indexing/atom_types-field*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP-group*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/energy_range-field*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/image_2d-group*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/image_2d/axis_i-field*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/image_2d/axis_i@long_name-attribute*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/image_2d/axis_i@units-attribute*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/image_2d/axis_j-field*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/image_2d/axis_j@long_name-attribute*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/image_2d/axis_j@units-attribute*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/image_2d/intensity-field*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/image_2d/intensity@units-attribute*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/image_2d/title-field*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/image_2d@axes-attribute*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/image_2d@AXISNAME_indices-attribute*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/image_2d@signal-attribute*
- */NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/iupac_line_candidates-field*
- */NXem/ENTRY/roiID/eds/indexing/summary-group*
- */NXem/ENTRY/roiID/eds/indexing/summary/axis_energy-field*
- */NXem/ENTRY/roiID/eds/indexing/summary/axis_energy@long_name-attribute*
- */NXem/ENTRY/roiID/eds/indexing/summary/intensity-field*
- */NXem/ENTRY/roiID/eds/indexing/summary/title-field*
- */NXem/ENTRY/roiID/eds/indexing/summary@axes-attribute*
- */NXem/ENTRY/roiID/eds/indexing/summary@AXISNAME_indices-attribute*
- */NXem/ENTRY/roiID/eds/indexing/summary@signal-attribute*
- */NXem/ENTRY/roiID/eels-group*
- */NXem/ENTRY/roiID/img-group*
- */NXem/ENTRY/roiID/img/imageID-group*
- */NXem/ENTRY/roiID/img/imageID/imaging_mode-field*
- */NXem/ENTRY/sample_reference_frame-group*
- */NXem/ENTRY/sample_reference_frame/alias-field*
- */NXem/ENTRY/sample_reference_frame/depends_on-field*
- */NXem/ENTRY/sample_reference_frame/origin-field*
- */NXem/ENTRY/sample_reference_frame/type-field*
- */NXem/ENTRY/sample_reference_frame/x-field*
- */NXem/ENTRY/sample_reference_frame/x_direction-field*

- */NXem/ENTRY/sample_reference_frame/y-field*
- */NXem/ENTRY/sample_reference_frame/y_direction-field*
- */NXem/ENTRY/sample_reference_frame/z-field*
- */NXem/ENTRY/sample_reference_frame/z_direction-field*
- */NXem/ENTRY/sampleID-group*
- */NXem/ENTRY/sampleID/atom_types-field*
- */NXem/ENTRY/sampleID/density-field*
- */NXem/ENTRY/sampleID/description-field*
- */NXem/ENTRY/sampleID/identifier_parent-field*
- */NXem/ENTRY/sampleID/identifier_parent@type-attribute*
- */NXem/ENTRY/sampleID/identifier_sample-field*
- */NXem/ENTRY/sampleID/identifier_sample@type-attribute*
- */NXem/ENTRY/sampleID/is_simulation-field*
- */NXem/ENTRY/sampleID/name-field*
- */NXem/ENTRY/sampleID/physical_form-field*
- */NXem/ENTRY/sampleID/preparation_date-field*
- */NXem/ENTRY/sampleID/thickness-field*
- */NXem/ENTRY/simulation-group*
- */NXem/ENTRY/simulation/config-group*
- */NXem/ENTRY/simulation/environment-group*
- */NXem/ENTRY/simulation/environment/PROGRAM-group*
- */NXem/ENTRY/simulation/environment/PROGRAM/program-field*
- */NXem/ENTRY/simulation/environment/PROGRAM/program@version-attribute*
- */NXem/ENTRY/simulation/programID-group*
- */NXem/ENTRY/simulation/programID/program-field*
- */NXem/ENTRY/simulation/programID/program@version-attribute*
- */NXem/ENTRY/simulation/results-group*
- */NXem/ENTRY/simulation/results/IMAGE-group*
- */NXem/ENTRY/simulation/results/interaction_volumeID-group*
- */NXem/ENTRY/simulation/results/interaction_volumeID/DATA-group*
- */NXem/ENTRY/simulation/results/interaction_volumeID/PROCESS-group*
- */NXem/ENTRY/simulation/results/SPECTRUM-group*
- */NXem/ENTRY/start_time-field*
- */NXem/ENTRY/userID-group*
- */NXem/ENTRY/userID/address-field*
- */NXem/ENTRY/userID/affiliation-field*

- */NXem/ENTRY/userID/email-field*
- */NXem/ENTRY/userID/identifierNAME-field*
- */NXem/ENTRY/userID/identifierNAME@type-attribute*
- */NXem/ENTRY/userID/name-field*
- */NXem/ENTRY/userID/role-field*
- */NXem/ENTRY/userID/telephone_number-field*
- */NXem/ENTRY/VARIADIC_reference_frame-group*
- */NXem/ENTRY/VARIADIC_reference_frame/alias-field*
- */NXem/ENTRY/VARIADIC_reference_frame/origin-field*
- */NXem/ENTRY/VARIADIC_reference_frame/type-field*
- */NXem/ENTRY/VARIADIC_reference_frame/x-field*
- */NXem/ENTRY/VARIADIC_reference_frame/x_direction-field*
- */NXem/ENTRY/VARIADIC_reference_frame/y-field*
- */NXem/ENTRY/VARIADIC_reference_frame/y_direction-field*
- */NXem/ENTRY/VARIADIC_reference_frame/z-field*
- */NXem/ENTRY/VARIADIC_reference_frame/z_direction-field*

NXDL Source:

<https://github.com/nexusformat/definitions/blob/main/applications/NXem.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 an indirect 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*

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*](#)

NXDL Source:

<https://github.com/nexusformat/definitions/blob/main/applications/NXiqproc.nxdl.xml>

NXlauetof**Status:**

application definition, extends *NXObject*

Description:

This is the application definition for a TOF laue diffractometer

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nXPixels: Number of X pixels

nYPixels: Number of Y pixels

nTOF: Time of flight

Groups cited:

NXdata, NXdetector, NXentry, NXinstrument, NXmonitor, NXsample

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: *NXlauetof*

instrument: (required) *NXinstrument* <=

detector: (required) *NXdetector* <=

This assumes a planar 2D detector. All angles and distances refer to the center of the detector.

polar_angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

The polar_angle (two theta) where the detector is placed.

azimuthal_angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

The azimuthal angle where the detector is placed.

data: (required) *NX_INT* (Rank: 3, Dimensions: [nXPixels, nYPixels, nTOF])

@signal: (required) *NX_POSINT*

Obligatory value: 1

x_pixel_size: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

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

distance: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

time_of_flight: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nTOF]) {units=*NX_TIME_OF_FLIGHT*} <=

sample: (required) *NXsample* <=

name: (required) *NX_CHAR* <=

Descriptive name of sample

orientation_matrix: (required) *NX_FLOAT* (Rank: 2, Dimensions: [3, 3]) <=

The orientation matrix according to Busing and Levy conventions. This is not strictly necessary as the UB can always be derived from the data. But let us bow to common usage which includes the UB nearly always.

unit_cell: (required) *NX_FLOAT* (Rank: 1, Dimensions: [6]) <=

The unit cell, a, b, c, alpha, beta, gamma. Again, not strictly necessary, but normally written.

control: (required) *NXmonitor* <=

mode: (required) *NX_CHAR* <=

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: `monitor` | `timer`

preset: (required) `NX_FLOAT`

preset value for time or monitor

data: (required) `NX_INT` (Rank: 1, Dimensions: [nTOF])

use these attributes `primary=1` `signal=1`

time_of_flight: (required) `NX_FLOAT` (Rank: 1, Dimensions: [nTOF])
`{units=NX_TIME_OF_FLIGHT} <=`

name: (required) `NXdata` <=

data: `link` (suggested target: `/NXentry/NXinstrument/NXdetector/data`)

time_of_flight: `link` (suggested target: `/NXentry/NXinstrument/NXdetector/time_of_flight`)

Hypertext Anchors

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

- `/NXlauetof/ENTRY-group`
- `/NXlauetof/ENTRY/control-group`
- `/NXlauetof/ENTRY/control/data-field`
- `/NXlauetof/ENTRY/control mode-field`
- `/NXlauetof/ENTRY/control/preset-field`
- `/NXlauetof/ENTRY/control/time_of_flight-field`
- `/NXlauetof/ENTRY/definition-field`
- `/NXlauetof/ENTRY/instrument-group`
- `/NXlauetof/ENTRY/instrument/detector-group`
- `/NXlauetof/ENTRY/instrument/detector/azimuthal_angle-field`
- `/NXlauetof/ENTRY/instrument/detector/data-field`
- `/NXlauetof/ENTRY/instrument/detector/data@signal-attribute`
- `/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.

data_scaling_factor: (optional) *NX_NUMBER* <=

An optional scaling factor to apply to the values in `data`.

The elements in `data` are often stored as integers for efficiency reasons and need further correction, generating floats. The two fields `data_scaling_factor` and `data_offset` allow linear corrections using the following convention:

`corrected_data = (data + offset) * scaling_factor`

This formula will derive the corrected value, when necessary.

`data_scaling_factor` is sometimes known as gain and `data_offset` is sometimes known as pedestal or background, depending on the community.

Use these fields to specify constants that need to be applied to the data to correct it to physical values. For example, if the detector gain is 10 counts per photon and a constant background of 400 needs to be subtracted off the pixels, specify `data_scaling_factor` as 0.1 and `data_offset` as -400 to specify the required conversion from raw counts to corrected photons. It is implied processing software will apply these corrections on-the-fly during processing.

The rank of these fields should either be a single value for the full dataset, a single per-pixel array applied to every image (dimensions (i, j) or (i, j, k)), or a per-image correction specified with an array whose slowest rank is nP (dimensions (np, 1), (np, i, j) or (np, i, j, k)).

When omitted, the scaling factor is assumed to be 1.

data_offset: (optional) `NX_NUMBER` <=

An optional offset to apply to the values in data.

When omitted, the offset is assumed to be 0.

See `data_scaling_factor` for more information.

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` \leq

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]) \leq

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` \leq

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` \leq

How many bits the electronics record per pixel.

detector_readout_time: (optional) `NX_FLOAT` {units=`NX_TIME`} \leq

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`} \leq

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`: (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`.

`BEAM`: (required) `NXbeam` <=

`@flux`: (optional) `NX_CHAR`

Which field contains the measured flux. One of `flux`, `total_flux`, `flux_integrated`, or `total_flux_integrated`.

Alternatively, the name being indicated could be a link to a dataset in an `NXmonitor` group that records per shot beam data.

`incident_wavelength`: (required) `NX_FLOAT` {units=`NX_WAVELENGTH`}
<=

In the case of a monochromatic beam this is the scalar wavelength.

Several other use cases are permitted, depending on the presence or absence of other `incident_wavelength_X` fields.

In the case of a polychromatic beam this is an array of length **m** of wavelengths, with the relative weights in `incident_wavelength_weights`.

In the case of a monochromatic beam that varies shot- to-shot, this is an array of wavelengths, one for each recorded shot. Here, `incident_wavelength_weights` and `incident_wavelength_spread` are not set.

In the case of a polychromatic beam that varies shot-to- shot, this is an array of length **m** with the relative weights in `incident_wavelength_weights` as a 2D array.

In the case of a polychromatic beam that varies shot-to- shot and where the channels also vary, this is a 2D array of dimensions **nP** by **m** (slow to fast) with the relative weights in `incident_wavelength_weights` as a 2D array.

Note, *variants* are a good way to represent several of these use cases in a single dataset, e.g. if a calibrated, single-value wavelength value is available along with the original spectrum from which it was calibrated.

incident_wavelength_weight: (optional) *NX_FLOAT*

DEPRECATED: use `incident_wavelength_weights`, see <https://github.com/nexusformat/definitions/issues/837>

In the case of a polychromatic beam this is an array of length **m** of the relative weights of the corresponding wavelengths in `incident_wavelength`.

In the case of a polychromatic beam that varies shot-to- shot, this is a 2D array of dimensions **nP** by **m** (slow to fast) of the relative weights of the corresponding wavelengths in `incident_wavelength`.

incident_wavelength_weights: (optional) *NX_FLOAT* <=

In the case of a polychromatic beam this is an array of length **m** of the relative weights of the corresponding wavelengths in `incident_wavelength`.

In the case of a polychromatic beam that varies shot-to- shot, this is a 2D array of dimensions **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/DATA/data_offset-field*
- */NXmx/ENTRY/DATA/data_scaling_factor-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*

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*

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*

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 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

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/ngu-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.

- /NXtofpd/ENTRY-group
- /NXtofpd/ENTRY/data-group
- /NXtofpd/ENTRY/data/data-link
- /NXtofpd/ENTRY/data/detector_number-link
- /NXtofpd/ENTRY/data/time_of_flight-link
- /NXtofpd/ENTRY/definition-field
- /NXtofpd/ENTRY/INSTRUMENT-group
- /NXtofpd/ENTRY/INSTRUMENT/detector-group
- /NXtofpd/ENTRY/INSTRUMENT/detector/azimuthal_angle-field
- /NXtofpd/ENTRY/INSTRUMENT/detector/data-field
- /NXtofpd/ENTRY/INSTRUMENT/detector/detector_number-field
- /NXtofpd/ENTRY/INSTRUMENT/detector/distance-field
- /NXtofpd/ENTRY/INSTRUMENT/detector/polar_angle-field
- /NXtofpd/ENTRY/INSTRUMENT/detector/time_of_flight-field
- /NXtofpd/ENTRY/MONITOR-group
- /NXtofpd/ENTRY/MONITOR/data-field
- /NXtofpd/ENTRY/MONITOR/distance-field
- /NXtofpd/ENTRY/MONITOR mode-field
- /NXtofpd/ENTRY/MONITOR/preset-field

- */NXtofpd/ENTRY/MONITOR/time_of_flight-field*
- */NXtofpd/ENTRY/pre_sample_fighthpath-field*
- */NXtofpd/ENTRY/SAMPLE-group*
- */NXtofpd/ENTRY/SAMPLE/name-field*
- */NXtofpd/ENTRY/start_time-field*
- */NXtofpd/ENTRY/title-field*
- */NXtofpd/ENTRY/user-group*
- */NXtofpd/ENTRY/user/name-field*

NXDL Source:

<https://github.com/nexusformat/definitions/blob/main/applications/NXtofpd.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_fighthpath: (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*

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.

- [*/NXzas/ENTRY-group*](#)
- [*/NXzas/ENTRY/DATA-group*](#)
- [*/NXzas/ENTRY/DATA/absorbed_beam-link*](#)
- [*/NXzas/ENTRY/DATA/energy-link*](#)
- [*/NXzas/ENTRY/DATA mode-field*](#)
- [*/NXzas/ENTRY/definition-field*](#)
- [*/NXzas/ENTRY/INSTRUMENT-group*](#)
- [*/NXzas/ENTRY/INSTRUMENT/absorbed_beam-group*](#)
- [*/NXzas/ENTRY/INSTRUMENT/absorbed_beam/data-field*](#)
- [*/NXzas/ENTRY/INSTRUMENT/incoming_beam-group*](#)
- [*/NXzas/ENTRY/INSTRUMENT/incoming_beam/data-field*](#)
- [*/NXzas/ENTRY/INSTRUMENT/monochromator-group*](#)
- [*/NXzas/ENTRY/INSTRUMENT/monochromator/energy-field*](#)
- [*/NXzas/ENTRY/INSTRUMENT/SOURCE-group*](#)
- [*/NXzas/ENTRY/INSTRUMENT/SOURCE/name-field*](#)
- [*/NXzas/ENTRY/INSTRUMENT/SOURCE/probe-field*](#)
- [*/NXzas/ENTRY/INSTRUMENT/SOURCE/type-field*](#)
- [*/NXzas/ENTRY/MONITOR-group*](#)
- [*/NXzas/ENTRY/MONITOR/data-field*](#)
- [*/NXzas/ENTRY/MONITOR mode-field*](#)
- [*/NXzas/ENTRY/MONITOR/preset-field*](#)
- [*/NXzas/ENTRY/SAMPLE-group*](#)
- [*/NXzas/ENTRY/SAMPLE/name-field*](#)
- [*/NXzas/ENTRY/start_time-field*](#)
- [*/NXzas/ENTRY/title-field*](#)

NXDL Source:

<https://github.com/nexusformat/definitions/blob/main/applications/NXzas.nxdl.xml>

NXzasproc

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*

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*

NXDL Source:

<https://github.com/nexusformat/definitions/blob/main/applications/NXxasproc.nxdl.xml>

NXxbase**Status:**

application definition, extends *NXObject*

Description:

This definition covers the common parts of all monochromatic single crystal raw data application definitions.

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

nXPixels: Number of X pixels

nYPixels: Number of Y pixels

Groups cited:

NXdata, *NXdetector*, *NXentry*, *NXinstrument*, *NXmonitor*, *NXmonochromator*, *NXsample*, *NXsource*

Structure:

ENTRY: (required) *NXentry*

title: (required) *NX_CHAR* <=

start_time: (required) *NX_DATE_TIME* <=

definition: (required) *NX_CHAR* <=

 Official NeXus NXDL schema to which this file conforms

 Obligatory value: NXxbase

instrument: (required) *NXinstrument* <=

source: (required) *NXsource* <=

type: (required) *NX_CHAR* <=

name: (required) *NX_CHAR* <=

probe: (required) *NX_CHAR* <=

 Any of these values: neutron | x-ray | electron

monochromator: (required) *NXmonochromator* <=

wavelength: (required) *NX_FLOAT* {units=*NX_WAVELENGTH*} <=

detector: (required) *NXdetector* <=

The name of the group is detector if there is only one detector, if there are several, names have to be detector1, detector2, ... detectorn.

data: (required) *NX_INT* (Rank: 3, Dimensions: [nP, nXPixels, nYPixels])

The area detector data, the first dimension is always the number of scan points, the second and third are the number of pixels in x and y. The origin is always assumed to be in the center of the detector. maxOccurs is limited to the the number of detectors on your instrument.

@signal: (required) *NX_POSINT*

Obligatory value: 1

x_pixel_size: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

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

distance: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

frame_start_number: (required) *NX_INT* <=

This is the start number of the first frame of a scan. In PX one often scans a couple of frames on a give sample, then does something else, then returns to the same sample and scans some more frames. Each time with a new data file. This number helps concatenating such measurements.

sample: (required) *NXsample* <=

name: (required) *NX_CHAR* <=

Descriptive name of sample

orientation_matrix: (required) *NX_FLOAT* (Rank: 2, Dimensions: [3, 3]) <=

The orientation matrix according to Busing and Levy conventions. This is not strictly necessary as the UB can always be derived from the data. But let us bow to common usage which includes the UB nearly always.

unit_cell: (required) *NX_FLOAT* (Rank: 1, Dimensions: [6]) <=

The unit cell, a, b, c, alpha, beta, gamma. Again, not strictly necessary, but normally written.

temperature: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) <=

The sample temperature or whatever sensor represents this value best

x_translation: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

Translation of the sample along the X-direction of the laboratory coordinate system

y_translation: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Translation of the sample along the Y-direction of the laboratory coordinate system

distance: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

Translation of the sample along the Z-direction of the laboratory coordinate system

control: (required) *NXmonitor* <=

mode: (required) *NX_CHAR* <=

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: **monitor | timer**

preset: (required) *NX_FLOAT*

preset value for time or monitor

integral: (required) *NX_FLOAT* {units=*NX_ANY*}

Total integral monitor counts

DATA: (required) *NXdata* <=

The name of this group id data if there is only one detector; if there are several the names will be data1, data2, data3 and will point to the corresponding detector groups in the instrument hierarchy. **data:** *link* (suggested target: /NXentry/NXinstrument/NXdetector/data)

Hypertext Anchors

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

- */NXbase/ENTRY-group*
- */NXbase/ENTRY/control-group*
- */NXbase/ENTRY/control/integral-field*
- */NXbase/ENTRY/control mode-field*
- */NXbase/ENTRY/control/preset-field*
- */NXbase/ENTRY/DATA-group*
- */NXbase/ENTRY/DATA/data-link*
- */NXbase/ENTRY/definition-field*
- */NXbase/ENTRY/instrument-group*
- */NXbase/ENTRY/instrument/detector-group*
- */NXbase/ENTRY/instrument/detector/data-field*
- */NXbase/ENTRY/instrument/detector/data@signal-attribute*
- */NXbase/ENTRY/instrument/detector/distance-field*
- */NXbase/ENTRY/instrument/detector/frame_start_number-field*
- */NXbase/ENTRY/instrument/detector/x_pixel_size-field*
- */NXbase/ENTRY/instrument/detector/y_pixel_size-field*
- */NXbase/ENTRY/instrument/monochromator-group*
- */NXbase/ENTRY/instrument/monochromator/wavelength-field*
- */NXbase/ENTRY/instrument/source-group*
- */NXbase/ENTRY/instrument/source/name-field*
- */NXbase/ENTRY/instrument/source/probe-field*

- */NXbase/ENTRY/instrument/source/type-field*
- */NXbase/ENTRY/sample-group*
- */NXbase/ENTRY/sample/distance-field*
- */NXbase/ENTRY/sample/name-field*
- */NXbase/ENTRY/sample/orientation_matrix-field*
- */NXbase/ENTRY/sample/temperature-field*
- */NXbase/ENTRY/sample/unit_cell-field*
- */NXbase/ENTRY/sample/x_translation-field*
- */NXbase/ENTRY/sample/y_translation-field*
- */NXbase/ENTRY/start_time-field*
- */NXbase/ENTRY/title-field*

NXDL Source:

<https://github.com/nexusformat/definitions/blob/main/applications/NXxeuler.nxdl.xml>

NXxeuler**Status:**

application definition, extends *NXbase*

Description:

raw data from a four-circle diffractometer with an eulerian cradle, extends *NXbase*

It extends *NXbase*, so the full definition is the content of *NXbase* 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 *NXbase*

Description:

raw data from a kappa geometry (CAD4) single crystal diffractometer, extends [NXbase](#)

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

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:

[NXdata](#), [NXdetector](#), [NXentry](#), [NXinstrument](#), [NXsample](#)

Structure:

ENTRY: (required) [NXentry](#) <=

definition: (required) [NX_CHAR](#) <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: [NXkappa](#)

instrument: (required) [NXinstrument](#) <=

detector: (required) [NXdetector](#) <=

polar_angle: (required) [NX_FLOAT](#) (Rank: 1, Dimensions: [nP])
{units=[NX_ANGLE](#)} <=

The polar_angle (two theta) at each scan point

sample: (required) [NXsample](#) <=

rotation_angle: (required) [NX_FLOAT](#) (Rank: 1, Dimensions: [nP])
{units=[NX_ANGLE](#)} <=

This is an array holding the sample rotation angle at each scan point

kappa: (required) [NX_FLOAT](#) (Rank: 1, Dimensions: [nP]) {units=[NX_ANGLE](#)}

This is an array holding the kappa angle at each scan point

phi: (required) [NX_FLOAT](#) (Rank: 1, Dimensions: [nP]) {units=[NX_ANGLE](#)}

This is an array holding the phi angle at each scan point

alpha: (required) [NX_FLOAT](#) {units=[NX_ANGLE](#)}

This holds the inclination angle of the kappa arm.

name: (required) [NXdata](#) <=

polar_angle: [link](#) (suggested target: /NXentry/NXinstrument/NXdetector/
polar_angle)

rotation_angle: [link](#) (suggested target: /NXentry/NXsample/rotation_angle)

kappa: [link](#) (suggested target: /NXentry/NXsample/kappa)

phi: [link](#) (suggested target: /NXentry/NXsample/phi)

Hypertext Anchors

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

- [/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.

- */NXlaugeplate/ENTRY-group*
- */NXlaugeplate/ENTRY/definition-field*
- */NXlaugeplate/ENTRY/instrument-group*
- */NXlaugeplate/ENTRY/instrument/detector-group*
- */NXlaugeplate/ENTRY/instrument/detector/diameter-field*

NXDL Source:

<https://github.com/nexusformat/definitions/blob/main/applications/NXlaugeplate.nxdl.xml>

NXnb

Status:

application definition, extends *NXbase*

Description:

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

This is the application definition for raw data from a single crystal diffractometer measuring in normal beam mode. It extends *NXbase*, so the full definition is the content of *NXbase* 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: NXnb

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.

- */NXnb/ENTRY-group*
- */NXnb/ENTRY/definition-field*
- */NXnb/ENTRY/instrument-group*
- */NXnb/ENTRY/instrument/detector-group*
- */NXnb/ENTRY/instrument/detector/polar_angle-field*
- */NXnb/ENTRY/instrument/detector/tilt_angle-field*
- */NXnb/ENTRY/name-group*
- */NXnb/ENTRY/name/polar_angle-link*
- */NXnb/ENTRY/name/rotation_angle-link*
- */NXnb/ENTRY/name/tilt-link*
- */NXnb/ENTRY/sample-group*
- */NXnb/ENTRY/sample/rotation_angle-field*

NXDL Source:

<https://github.com/nexusformat/definitions/blob/main/applications/NXnb.nxdl.xml>

NXxrot

Status:

application definition, extends *NXbase*

Description:

raw data from a rotation camera, extends *NXbase*

This is the application definition for raw data from a rotation camera. It extends *NXbase*, so the full definition is the content of *NXbase* plus the data defined here.

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:

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:

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.

NXchamber

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

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

Base class to hold different coordinate systems and representation conversions.

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.

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.

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

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.

NXenergydispersion

Subclass of NXelectronanalyser to describe the energy dispersion section of a

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.

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_opt

Description of an optical lens.

NXmagnetic_kicker

definition for a magnetic kicker.

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.

NXorientation_set

Details about individual orientations of a set of objects.

NXpolarizer_opt

An optical polarizer.

NXpulser_apm

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

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*.

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

NXspin_rotator

definition for a spin rotator.

NXspindispersion

Subclass of NXelectronanalyser to describe the spin filters in photoemission

NXsubsampling_filter

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

NXsubstance

A form of matter with a constant, definite chemical composition.

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 with a focus on EM:

NXaberration and :ref:NXcorrector_cs`:

Base class to describe procedures and values for the calibration of (spherical) aberrations and the devices for achieving this technically.

NXcoordinate_system:

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.

NXbeam_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.

NXbeam_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:

Base class 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.

NXinstrument_em:

A base class which defines all modular parts that make up an instrument (real or simulated) for studying electron matter interaction. This base class is used in NXem in two places: One that is placed inside an ENTRY.measurement.instrument group. This group holds all those (meta)data which do not change during a session, i.e. instrument name, typically identifier of hardware components or version of control software. Another one that is placed inside an ENTRY.measurements.events group. This group holds all those (meta)data data change when collecting data during a session.

NXatom and 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](https://doi.org/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:

Base class and specializations comparable to NXimage 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.

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, NXem_eds, NXem_eels, NXem_img:

Base class providing concepts for specific data acquisition modes and associated analysis used in electron microscopy such as collecting and indexing Kikuchi pattern into orientation maps for the two-dimensional, three-, X-ray spectroscopy, different imaging modes, or electron energy loss spectroscopy (EELS).

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 one 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 separately 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 $\text{eps} = \text{eps_inf} + \sum[\text{A} * \lambda^{** 2} / (\lambda^{** 2} - \text{B}^{** 2})]$ (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 grammar 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
```

(continues on next page)

(continued from previous page)

```
?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

?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 photoemission experiments.

Base Classes

NXelectronanalyser:

A base class to describe electron kinetic energy analyzers. 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 system (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 an 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:

NXcoordinate_system:

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

NXatom and NXion:

Base classes to describe groups of atoms and 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](https://doi.org/10.1017/S1431927621012241) are used to map isotope to hash values with which all possible isotopes can be described.

NXchemical_composition:

Base class to report the chemical composition of a sample or its parts.

NXcircuit:

Base class to describe electronic circuits.

NXroi:

Base class to report details about results obtained for a specific region of a sample a region-of-interest.

NXcomponent and NXfabrication:

Base classes to group frequently used descriptions such as physical parts an is constructed from instrument and manufacturing details of it bundling manufacturer/technology-partner-specific details.

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.

NXmanipulator:

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.

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/trajecory. 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.

NXisocountour:

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.

NXapm

Status:

application definition (contribution), 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_ivc_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, NXbeam, NXchamber, NXchemical_composition, NXcollection, NXcoordinate_system_set, NXcsg, NXdata, NXdetector, NXentry, NXfabrication, NXinstrument, NXion, NXlens_em, NXmanipulator, NXmonitor, NXnote, NXpeak, NXprocess, NXprogram, NXpulser_apm, NXpump, NXreflectron, NXsample, NXsource, 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_poly-crystalline: (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: (optional) *NXaperture*

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) *NXmanipulator*

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 frustum 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) (AP-Suite/IVAS/DAVis).

By contrast, instruments which were built by individual research groups such as of the French (GPM, Rouen, France), the Schmitz (Inspico, Stuttgart, Germany), the Felfer (Oxcart, Erlangen, Germany), the Northwestern (D. Isheim, Seidman group et al.), or the PNNL group (Pacific Northwest National Laboratory, Portland, Oregon, U.S.) have other solutions to control the instrument.

Some of which are modularized and open, some of which realize also integrated control units with portions of eventually undisclosed source code and (so far) lacking (support of)/open APIs.

Currently, there is no accepted/implemented community-specific API for getting finely granularized access to such control settings.

These considerations motivated the design of the NXapm application definition in that it stores quantities in NXcollection. groups to begin with. Holding heterogeneous, not yet standardized but relevant pieces of information is the purpose of this collection. 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 of 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.

xmdf_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*

xmdf_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-group
- /NXapm/ENTRY/atom_probe/local_electrode/APERTURE/FABRICATION-group
- /NXapm/ENTRY/atom_probe/local_electrode/APERTURE/FABRICATION/capabilities-field
- /NXapm/ENTRY/atom_probe/local_electrode/APERTURE/FABRICATION/identifier-field
- /NXapm/ENTRY/atom_probe/local_electrode/APERTURE/name-field
- /NXapm/ENTRY/atom_probe/local_electrode/APERTURE/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/PROGRAM/program@version-attribute*
- */NXapm/ENTRY/run_number-field*
- */NXapm/ENTRY/sample-group*
- */NXapm/ENTRY/sample/CHEMICAL_COMPOSITION-group*

- */NXapm/ENTRY/sample/CHEMICAL_COMPOSITION/ION-group*
- */NXapm/ENTRY/sample/CHEMICAL_COMPOSITION/ION/composition-field*
- */NXapm/ENTRY/sample/CHEMICAL_COMPOSITION/ION/composition_error-field*
- */NXapm/ENTRY/sample/CHEMICAL_COMPOSITION/ION/name-field*
- */NXapm/ENTRY/sample/CHEMICAL_COMPOSITION/normalization-field*
- */NXapm/ENTRY/sample/description-field*
- */NXapm/ENTRY/sample/grain_diameter-field*
- */NXapm/ENTRY/sample/grain_diameter_error-field*
- */NXapm/ENTRY/sample/heat_treatment_quenching_rate-field*
- */NXapm/ENTRY/sample/heat_treatment_quenching_rate_error-field*
- */NXapm/ENTRY/sample/heat_treatment_temperature-field*
- */NXapm/ENTRY/sample/heat_treatment_temperature_error-field*
- */NXapm/ENTRY/specimen-group*
- */NXapm/ENTRY/specimen/alias-field*
- */NXapm/ENTRY/specimen/atom_types-field*
- */NXapm/ENTRY/specimen/description-field*
- */NXapm/ENTRY/specimen/is_polycrystalline-field*
- */NXapm/ENTRY/specimen/name-field*
- */NXapm/ENTRY/specimen/preparation_date-field*
- */NXapm/ENTRY/specimen/sample_history-field*
- */NXapm/ENTRY/start_time-field*
- */NXapm/ENTRY/thumbnail-group*
- */NXapm/ENTRY/thumbnail@type-attribute*
- */NXapm/ENTRY/USER-group*
- */NXapm/ENTRY/USER/address-field*
- */NXapm/ENTRY/USER/affiliation-field*
- */NXapm/ENTRY/USER/email-field*
- */NXapm/ENTRY/USER/name-field*
- */NXapm/ENTRY/USER/orcid-field*
- */NXapm/ENTRY/USER/orcid_platform-field*
- */NXapm/ENTRY/USER/role-field*
- */NXapm/ENTRY/USER/social_media_name-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 (contribution), 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/capabilities-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_size-field*
- */NXapm_composition_space_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_size-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 (contribution), 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 (contribution), 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 (contribution), 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_ivemax: 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 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_ivectmax]) {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, num=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 (contribution), 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 form 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 further 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_normal-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 (contribution), 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-degenerated 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 inferred 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 inferred 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/filename@attribute
- /NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/current_set/FEATURE/groupname@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
- /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_type@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/groupname field*
- */NXapm_paraprobe_config_intersector/ENTRY/VOLUME_VOLUME_SPATIAL_CORRELATION/next_set/number_of_feature_type 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 (contribution), 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 isotope_whitelist.
- composition, candidates but normalized by composition, i.e. at-%
- concentration, candidates but normalized by voxel volume, i.e. ions/nm³

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. Reichenmann 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 smoothed.

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 unrange 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/delocalization/isosurfacing/has_object_properties-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/has_proxy-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/has_proxy_edge_contact-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/has_proxy_geometry-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/has_proxy_ions-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/has_proxy_obb-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/has_proxy_properties-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/has_triangle_soup-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isosurfacing/phi-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/isotope_whitelist-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/kernel_size-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/kernel_variance-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/delocalization/normalization-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/edge_of_the_dataset-group
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/edge_of_the_dataset/dataset_name_facet_indices-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/edge_of_the_dataset/dataset_name_vertices-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/edge_of_the_dataset/filename-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/edge_of_the_dataset/filename@version-attribute
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/evaporation_id_filter-group
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/hit_multiplicity_filter-group
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/interface_meshing-group
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/interface_meshing/control_points-field
- /NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/interface_meshing/decorating_iontypes_filter-group

- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/interface_meshing/decorating_iontypes_filter/candidates-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/interface_meshing/filename-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/interface_meshing/filename@version-attribute*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/interface_meshing/initialization-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/interface_meshing/method-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/interface_meshing/number_of_iterations-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/interface_meshing/target_dcom_radius-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/interface_meshing/target_edge_length-field*
- */NXapm_paraprobe_config_nanochem/ENTRY/PROCESS/interface_meshing/target_smoothing_step-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 (contribution), 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. Relativistic 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/dimensionality-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_offset-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/dimensionality-field

- /NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_ELLIPSOID_SET/half_axes_radius-field
- /NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_ELLIPSOID_SET/identifier_offset-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/cardinality-field
- /NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_HEXAHEDRON_SET/dimension-field
- /NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CG_HEXAHEDRON_SET/hexahedron-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/bitfield
- /NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CS_FILTER_BOOLEAN_MASK/ideal-field
- /NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CS_FILTER_BOOLEAN_MASK/match-field
- /NXapm_paraprobe_config_ranger/ENTRY/PROCESS/apply_existent_ranging/spatial_filter/CS_FILTER_BOOLEAN_MASK/number-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 (contribution), 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 (contribution), 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_ivectmax: 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 MT19937 (MersenneTwister) pseudo-random number generator to shuffle the ion-type 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 ion-type, 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_ivectmax]) {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_ivemax]) {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_offset-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/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 (contribution), 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 (contribution), 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 (contribution), 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 (contribution), 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/capabilities-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_size-field`
- `/NXapm_paraprobe_results_clusterer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_size-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 (contribution), 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.

xmf_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/capabilities-field
- /NXapm_paraprobe_results_distancer/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/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_size-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 (contribution), 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 precipitates 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/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/capabilities-field*
- */NXapm_paraprobe_results_intersector/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/identifier-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_and_group
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/coprecipitation_and_field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/coprecipitation_and_field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/coprecipitation_and_field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/coprecipitation_and_field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/coprecipitation_and_field
- /NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/current_to_next_line-field

- */NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/current_to_next_line-field*
- */NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/intersection_volume-field*
- */NXapm_paraprobe_results_intersector/ENTRY/PROCESS/VOLUME_VOLUME_SPATIAL_CORRELATION/next_to_current_line-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 (contribution), 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 $2 * \text{kernel_extent}[i] + 1$ 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, summing 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: $[\$sum_i = 0]^{\{i = n-1\}}$, $\$sum_{\{i=0\}}^{\{i = n\}}$].$

xmdf_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.

xmdf_intensity: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_xyz]) {units=*NX_ANY*}

Intensity of the field at given point

xmdf_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 XDFM 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, summing 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: $[\$sum_i = 0]^{\{i = n-1\}}, \$sum_{\{i=0\}}^{\{i = n\}}$.

xmf_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: (re-
 quired) *NX_UINT* (Rank: 1, Dimen-
 sions: [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 sur-
 face of the feature.

proxies: (optional) *NXprocess*

Details for features which are so-called proxies, i.e.
 objects which have been reconstructed and combi-
 natorially closed with processing their partial tri-
 angulated_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, Dimen-
 sions: [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 fea-
 ture_identifier.

feature_identifier: (required)
NX_UINT (Rank: 1, Dimensions: [i])
 {units=*NX_UNITLESS*}

volume: (required) *NX_FLOAT* (Rank: 1, Di-
 mensions: [i]) {units=*NX_VOLUME*}

composition: (optional) *NXchemi-
 cal_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: (re-
quired) *NX_UINT* (Rank: 1, Dimen-
sions: [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 sur-
face 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 dis-
tant 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, Di-
mensions: [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)	<i>NX_INT</i>
{units= <i>NX_UNITLESS</i> } <=		
face_identifier_offset:	(required)	<i>NX_INT</i>
{units= <i>NX_UNITLESS</i> } <=		
face_identifier: (required) <i>NX_UINT</i> (Rank: 1, Dimensions: [j])		
{units= <i>NX_UNITLESS</i> }		
vertices: (required) <i>NX_NUMBER</i> (Rank: 2, Dimensions: [i, 3])		
{units= <i>NX_LENGTH</i> } <=		
faces: (required) <i>NX_UINT</i> (Rank: 2, Dimensions: [j, 3])		
{units= <i>NX_UNITLESS</i> }		
xdmf_topology: (required) <i>NX_UINT</i> (Rank: 1, Dimensions: [k])		
{units= <i>NX_UNITLESS</i> }		
face_normal: (required) <i>NXcg_unit_normal_set</i> <=		
dimensionality: (required) <i>NX_POSINT</i> {units= <i>NX_UNITLESS</i> }		
<=		
cardinality: (required) <i>NX_POSINT</i> {units= <i>NX_UNITLESS</i> } <=		
normals: (required) <i>NX_FLOAT</i> (Rank: 2, Dimensions: [j, 3])		
{units= <i>NX_LENGTH</i> } <=		
Direction of each normal		
orientation: (required) <i>NX_UINT</i> (Rank: 1, Dimensions: [j])		
{units= <i>NX_UNITLESS</i> }		
Qualifier how which specifically oriented normal to its primitive each normal represents.		
• 0 - undefined		
• 1 - outer		
• 2 - inner		
vertex_normal: (required) <i>NXcg_unit_normal_set</i> <=		
dimensionality: (required) <i>NX_POSINT</i> {units= <i>NX_UNITLESS</i> }		
<=		
cardinality: (required) <i>NX_POSINT</i> {units= <i>NX_UNITLESS</i> } <=		
normals: (required) <i>NX_FLOAT</i> (Rank: 2, Dimensions: [j, 3])		
{units= <i>NX_LENGTH</i> } <=		
Direction of each normal		
orientation: (required) <i>NX_UINT</i> (Rank: 1, Dimensions: [j])		
{units= <i>NX_UNITLESS</i> }		
Qualifier how which specifically oriented normal to its primitive each normal represents.		
• 0 - undefined		
• 1 - outer		
• 2 - inner		
MESH_CURR_POST_DCOM_STEP: (optional) <i>NXcg_triangle_set</i>		

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*}

xmdf_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*

xmdf_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/capabilities-field`](#)
- [`/NXapm_paraprobe_results_nanochem/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identifier-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/dimensionality-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_normal-group*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/face_normal-field*
- */NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_POST_DCOM_STEP/face_normal-field*

- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/vertex_normal_group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/vertex_normal_field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/vertex_normal_field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/vertex_normal_field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/interface_modelling/MESH_CURR_PRE_DCOM_STEP/vertex_normal_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/boundary_field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/boundary_field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/hexahedron_group
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/hexahedron_field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/hexahedron_field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/hexahedron_field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/hexahedron_field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/hexahedron_field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/hexahedron_field
- /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/bounding_box/identification_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/extents-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/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_magnitude group
 - /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitude field
 - /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitude attribute
 - /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitude attribute
 - /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitude attribute
 - /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_magnitude attribute
 - /NXapm_paraprobe_results_nanochem/ENTRY/PROCESS/iso_surface_analysis/DELOCALIZATION/grid/scalar_field_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/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/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/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 (contribution), 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_ivect_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* <=br/>**role:** (recommended) *NX_CHAR* <=br/>**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 ion-type 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_snap-field*
- */NXapm_paraprobe_results_ranger/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_snap-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 (contribution), 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* <=br/>
role: (recommended) *NX_CHAR* <=br/>
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/capabilities-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_size-field*
- */NXapm_paraprobe_results_selector/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_size-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 (contribution), 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*, *NX-transformations*, *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/capabilities-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_size-field`
- `/NXapm_paraprobe_results_spatstat/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_size-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 (contribution), 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*}

xmf_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*}

xmf_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

ing each triangle. The total number of entries is n_f_tet
 \ast (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* \leqslant

command_line_call: (optional) *NX_CHAR* \leqslant

start_time: (recommended) *NX_DATE_TIME* \leqslant

end_time: (recommended) *NX_DATE_TIME* \leqslant

total_elapsed_time: (required) *NX_NUMBER* \leqslant

number_of_processes: (required) *NX_POSINT*

number_of_threads: (required) *NX_POSINT*

number_of_gpus: (required) *NX_POSINT*

CS_COMPUTER: (recommended) *NXcs_computer* \leqslant

name: (recommended) *NX_CHAR* \leqslant

operating_system: (required) *NX_CHAR* \leqslant

@version: (required) *NX_CHAR* \leqslant

uuid: (optional) *NX_CHAR* \leqslant

CS_CPU: (optional) *NXcs_cpu*

name: (optional) *NX_CHAR* \leqslant

FABRICATION: (recommended) *NXfabrication* \leqslant

identifier: (optional) *NX_CHAR*

capabilities: (optional) *NX_CHAR*

CS_GPU: (optional) *NXcs_gpu*

name: (optional) *NX_CHAR* \leqslant

FABRICATION: (recommended) *NXfabrication* \leqslant

identifier: (optional) *NX_CHAR*

capabilities: (optional) *NX_CHAR*

CS_MM_SYS: (optional) *NXcs_mm_sys*

total_physical_memory: (required) *NX_NUMBER* \leqslant

CS_IO_SYS: (optional) *NXcs_io_sys*

CS_IO_OBJ: (required) *NXcs_io_obj* \leqslant

technology: (required) *NX_CHAR* \leqslant

max_physical_capacity: (required) *NX_NUMBER* \leqslant

name: (optional) *NX_CHAR* \leqslant

FABRICATION: (recommended) *NXfabrication* \leqslant

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/group*

- */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/capabilities-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_size-field*
- */NXapm_paraprobe_results_surfacer/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_size-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/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/dimensions-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_id-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_type-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 (contribution), 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/polylines 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/capabilities-field*
- */NXapm_paraprobe_results_tessellator/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identifier-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 (contribution), 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 <=*

xmdf_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_ivc_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_pr-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/capabilities-field*
- */NXapm_paraprobe_results_transcoder/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/FABRICATION/identifier-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 (contribution), 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, describe 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 diffraction (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 (contribution), extends [NXcomponent](#)

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 describe 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 outgoing (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 geometry). For a homogeneous thickness (e.g. as in plate beam splitters) the minimum and maximum values are equal.

index_of_refraction_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 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_refraction_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

NXcg_alpha_complex

Status:

base class (contribution), 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 (contribution), 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 upper_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 (contribution), 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 (contribution), 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 savvy 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 used 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, summing 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 (contribution), 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 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 (contribution), 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 (contribution), extends *NXObject*

Description:

Computational geometry 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 (contribution), 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 (contribution), 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 (contribution), 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 through 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 parallelogram. 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 (contribution), 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 (contribution), 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/polylines-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 (contribution), 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 counted 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 (contribution), 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: $[\sum_{i=0}^{i=N-1}, \sum_{i=0}^N]$.

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 (contribution), 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 (contribution), 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 (contribution), 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 (contribution), 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) (optional) 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 (contribution), 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 (contribution), 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 (contribution), extends [NXcomponent](#)

Description:

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

Symbols:

No symbol table

Groups cited:

none

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.

Hypertext Anchors

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

- [/NXchamber/description-field](#)
- [/NXchamber/name-field](#)

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXchamber.nxdl.xml

NXclustering**Status:**

base class (contribution), 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 (contribution), extends [NXcomponent](#)

Description:

Subclass of NXelectronanalyser to describe the electron collection column of a photoelectron analyser.

Symbols:

No symbol table

Groups cited:

[NXaperture](#), [NXdeflector](#), [NXlens_em](#)

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.

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/extractor_current-field*
- */NXcollectioncolumn/extractor_voltage-field*
- */NXcollectioncolumn/LENS_EM-group*
- */NXcollectioncolumn/magnification-field*
- */NXcollectioncolumn mode-field*
- */NXcollectioncolumn/projection-field*
- */NXcollectioncolumn/scheme-field*
- */NXcollectioncolumn/working_distance-field*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcollectioncolumn.nxdl.xml

NXcontainer

Status:

base class (contribution), extends *NXcomponent*

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:

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:

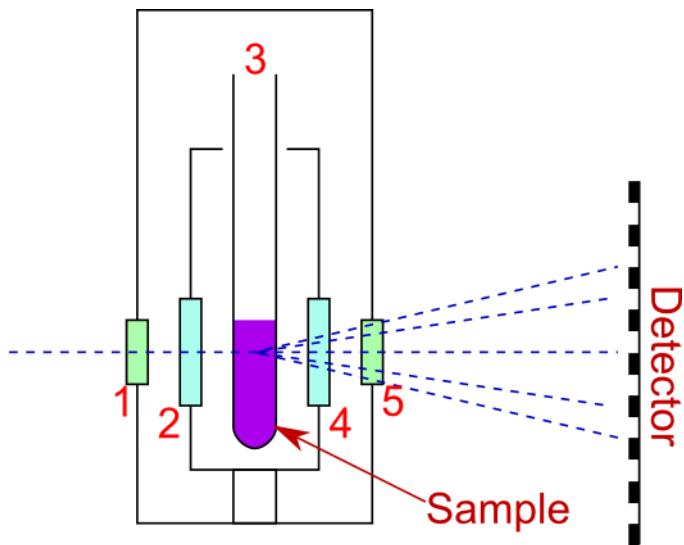


Fig. 12: A hypothetical capillary furnace. The beam passes from left to right (blue dashes), passing through window 1, then window 2, before passing through the downstream wall of the capillary. It is then scattered by the sample with scattered beams passing through the upstream wall of the capillary, 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.

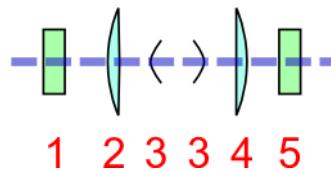


Fig. 13: A complete description of the shapes of the container elements with their orientation relative to the beam and also information on whether they are upstream or downstream of the sample is also therefore important. For example, although the windows 2 and 4 have the same shape, the path taken 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 (contribution), extends [NXobject](#)

Description:

Base class to hold different coordinate systems and representation conversions.

How many nodes of type [NXcoordinate_system_set](#) should be used in an application definition?

- 0; if there is no instance of [NXcoordinate_system_set](#) and therein or elsewhere across the application definition, an instance of NXcoordinate_system is defined, the default NeXus [McStas](#) coordinate system is assumed. This makes [NXcoordinate_system_set](#) and NXcoordinate_system base classes backwards compatible to older NeXus conventions and classes.
- 1; if only one [NXcoordinate_system_set](#) is defined, it should be placed as high up in the node hierarchy (ideally right below an instance of NXentry) of the application definition tree as possible. This [NXcoordinate_system_set](#) should define at least one NXcoordinate_system instance. This shall be named such that it is clear how this coordinate system is typically referred to in a community. For the NeXus [McStas](#) coordinate system, *it is advised to call it mcstas for the sake of improved clarity. Additional NXcoordinate_system instances should be specified if possible in that same :ref: 'NXcoordinate_system_set' instead of cluttering them across the tree.*

If this is the case, it is assumed that the NXcoordinate_system_members overwrite the NeXus default McStas coordinate system, i.e. users can thereby conveniently and explicitly specify the coordinate system(s) that they wish to use.

Users are encouraged to write also explicit and clean depends_on fields in all groups that encode information about where the interpretation of coordinate systems is relevant. If these depends_on hints are not provided, it is automatically assumed that all children (to arbitrary depth) of that branch and sub-branches below the one in which that [NXcoordinate_system_set](#) is defined use either the only

NXcoordinate_system_set instance in that set or the application definition is considered underconstrained which should at all costs be avoided and in which case again McStas is assumed.

- 2 and more; as soon as more than one *NXcoordinate_system_set* is specified somewhere in the tree, different interpretations are possible as to which of these coordinate system sets and instances apply or take preference. We realize that such ambiguities should at all costs be avoided. However, the opportunity for multiple sets and their instances enables to have branch-specific coordinate system conventions which could especially be useful for deep classes where multiple scientific methods are combined or cases where having a definition of global translation and conversion tables how to convert between representations in different coordinate systems is not desired or available for now. We argue that having 2 or more *NXcoordinate_system_set* instances and respective NXcoordinate_system instances makes the interpretation eventually unnecessary complicated. Instead, developers of application definitions should always try to work for clarity and thus use only one top-level coordinate system set.

For these reasons we conclude that the option with one top-level *NXcoordinate_system_set* instance is the preferred choice.

McStas is used if neither an instance of *NXcoordinate_system_set* nor an instance of NXcoordinate_system is specified. However, even in this case it is better to be explicit like for every other coordinate system definition to support users with interpreting the content and logic behind every instance of the tree.

How to store coordinate systems inside *NXcoordinate_system_set*? Individual coordinate systems should be specified as members of the *NXcoordinate_system_set* instance using instances of NXcoordinate_system.

How many individual instances of NXcoordinate_system to allow within one instance of *NXcoordinate_system_set*?

- 0; This case should be avoided for the sake of clarity but this case could mean the authors of the definition meant that McStas is used. We conclude, McStas is used in this case.
- 1; Like above-mentioned this case has the advantage that it is explicit and faces no ambiguities. However, in reality typically multiple coordinate systems have to be mastered especially for complex multi-signal modality experiments.
- 2 or more; If this case is realized, the best practice is that in every case where a coordinate system should be referred to the respective class has a depends_on field which resolves the possible ambiguities which specific coordinate systems is referred to. The benefit of this explicit and clear specifying of the coordinate system used in every case is that especially in combination with having coordinate systems inside deeper branches makes up for a very versatile, backwards compatible, but powerful system to express different types of coordinate systems using NeXus. In the case of two or more instances of NXcoordinate_system in one *NXcoordinate_system_set*, it is also advised to specify the relationship between the two coordinate systems by using the (NXtransformations) group within NXcoordinate_system.

In effect, 1 should be the preferred choice. However, if more than one coordinate system is defined for practical purposes, explicit depends_on fields should always guide the user for each group and field which of the coordinate system one refers to.

Symbols:

No symbol table

Groups cited:

NXcoordinate_system

Structure:

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. This is in accordance with convention 2 of reference <https://doi.org/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: `counter_clockwise | clockwise`

rotation_convention: (optional) *NX_CHAR*

How are rotations interpreted into an orientation according to convention 3 of reference <https://doi.org/10.1088/0965-0393/23/8/083501>.

Any of these values: `passive | active`

euler_angle_convention: (optional) *NX_CHAR*

How are Euler angles interpreted given that there are several choices (e.g. zxz, xyz) according to convention 4 of reference <https://doi.org/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. Apart from undefined, proper Euler angles are distinguished from (improper) Tait-Bryan angles.

Any of these values:

- zxz
- xyx
- yzy
- zyz
- xzx
- yxy
- xyz
- yzx
- zxy
- xzy
- zyx
- yxz

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 reference <https://doi.org/10.1088/0965-0393/23/8/083501>.

Obligatory value: `rotation_angle_on_interval_zero_to_pi`

sign_convention: (optional) *NX_CHAR*

Which sign convention is followed when converting orientations between different parameterizations/representations according to convention 6 of reference <https://doi.org/10.1088/0965-0393/23/8/083501>.

Any of these values: `p_plus_one | p_minus_one`

COORDINATE_SYSTEM: (optional) *NXcoordinate_system*

processing_reference_frame: (optional) *NXcoordinate_system*

Details about eventually relevant named directions that may give reasons for anisotropies. The classical example is mechanical processing where one has to specify which directions (e.g. rolling, transverse, and normal direction) align how with the direction of the base vectors of the sample_reference_frame.

It is assumed that the configuration is inspected by looking towards the sample surface. If a detector is involved, it is assumed that the configuration is inspected from a position that is located behind this detector.

If any of these assumptions is not met, the user is required to explicitly state this.

Reference <https://doi.org/10.1016/j.matchar.2016.04.008> suggests to label the base vectors of this coordinate system as Xp, Yp, Zp.

sample_reference_frame: (optional) *NXcoordinate_system*

Details about the sample_reference_frame that is typically overlaid onto the surface of the sample.

It is assumed that the configuration is inspected by looking towards the sample surface. If a detector is involved, it is assumed that the configuration is inspected from a position that is located behind this detector.

If any of these assumptions is not met, the user is required to explicitly state this.

Reference <https://doi.org/10.1016/j.matchar.2016.04.008> suggests to label the base vectors of this coordinate system as Xs, Ys, Zs.

DETECTOR_REFERENCE_FRAME: (optional) *NXcoordinate_system*

Details about the detector_reference_frame for a specific detector.

Reference <https://doi.org/10.1016/j.matchar.2016.04.008> suggests to label the base vectors of this coordinate system as Xd, Yd, Zd.

It is assumed that the configuration is inspected by looking towards the sample surface from a position that is located behind the detector.

If any of these assumptions is not met, the user is required to explicitly state this.

Instances should use detector_reference_frame as a name prefix.

Hypertext Anchors

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

- [*/NXcoordinate_system_set/axis_angle_convention-field*](#)
- [*/NXcoordinate_system_set/COORDINATE_SYSTEM-group*](#)
- [*/NXcoordinate_system_set/DETECTOR_REFERENCE_FRAME-group*](#)
- [*/NXcoordinate_system_set/euler_angle_convention-field*](#)
- [*/NXcoordinate_system_set/processing_reference_frame-group*](#)
- [*/NXcoordinate_system_set/rotation_convention-field*](#)
- [*/NXcoordinate_system_set/rotation_handedness-field*](#)
- [*/NXcoordinate_system_set/sample_reference_frame-group*](#)

- */NXcoordinate_system_set/sign_convention-field*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcoordinate_system_set.nxdl.xml

NXcs_cpu**Status:**

base class (contribution), 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 (contribution), 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 (contribution), 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 (contribution), 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 (contribution), 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 (contribution), 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 (contribution), 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

NXcsg

Status:

base class (contribution), 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_ptcho

Status:

application definition (contribution), 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 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](#), [NX-transformations](#)

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 “transla-
tion:\$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

NXdelocalization**Status:**

base class (contribution), 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 (contribution), 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 (contribution), 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 recommended 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 used 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 recommended 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 (contribution), 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 (contribution), 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 (contribution), 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 (contribution), 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 crystallin 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 diagonalized 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 diagonalized 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 diagonalized 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*

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_x/DISPERSION_FUNCTION/DISPERSION_SINGLE_PARAMETER/name-field*
- */NXdispersive_material/ENTRY/dispersion_x/DISPERSION_FUNCTION/DISPERSION_SINGLE_PARAMETER/value-field*
- */NXdispersive_material/ENTRY/dispersion_x/DISPERSION_FUNCTION/energy_identifier-field*

- `/NXdispersive_material/ENTRY/dispersion_x/DISPERSION_FUNCTION/energy_unit-field`
- `/NXdispersive_material/ENTRY/dispersion_x/DISPERSION_FUNCTION/formula-field`
- `/NXdispersive_material/ENTRY/dispersion_x/DISPERSION_FUNCTION/model_name-field`
- `/NXdispersive_material/ENTRY/dispersion_x/DISPERSION_FUNCTION/representation-field`
- `/NXdispersive_material/ENTRY/dispersion_x/DISPERSION_FUNCTION/wavelength_identifier-field`
- `/NXdispersive_material/ENTRY/dispersion_x/DISPERSION_FUNCTION/wavelength_unit-field`
- `/NXdispersive_material/ENTRY/dispersion_x/DISPERSION_TABLE-group`
- `/NXdispersive_material/ENTRY/dispersion_x/DISPERSION_TABLE/convention-field`
- `/NXdispersive_material/ENTRY/dispersion_x/DISPERSION_TABLE/dielectric_function-field`
- `/NXdispersive_material/ENTRY/dispersion_x/DISPERSION_TABLE/model_name-field`
- `/NXdispersive_material/ENTRY/dispersion_x/DISPERSION_TABLE/refractive_index-field`
- `/NXdispersive_material/ENTRY/dispersion_x/DISPERSION_TABLE/wavelength-field`
- `/NXdispersive_material/ENTRY/dispersion_x/model_name-field`
- `/NXdispersive_material/ENTRY/dispersion_x/plot-group`
- `/NXdispersive_material/ENTRY/dispersion_y-group`
- `/NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION-group`
- `/NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/convention-field`
- `/NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/DISPERSION_REPEAT_PARAMETER-group`
- `/NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/DISPERSION_REPEAT_PARAMETER/name-field`
- `/NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/DISPERSION_REPEAT_PARAMETER/values-field`
- `/NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/DISPERSION_SINGLE_PARAMETER-group`
- `/NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/DISPERSION_SINGLE_PARAMETER/name-field`
- `/NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/DISPERSION_SINGLE_PARAMETER/value-field`
- `/NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/energy_identifier-field`
- `/NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/energy_unit-field`
- `/NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/formula-field`
- `/NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/model_name-field`
- `/NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/representation-field`
- `/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_REPEATABLE_PARAMETER-group`
- `/NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/DISPERSION_REPEATABLE_PARAMETER/name-field`
- `/NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/DISPERSION_REPEATABLE_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

NXelectronanalyser**Status:**

base class (contribution), extends *NXcomponent*

Description:

Subclass of NXinstrument to describe a photoelectron analyser.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays

nfa: Number of fast axes (axes acquired simultaneously, without scanning a physical quantity)

nsa: Number of slow axes (axes acquired scanning a physical quantity)

Groups cited:

NXcollectioncolumn, *NXdeflector*, *NXdetector*, *NXenergydispersion*, *NXlens_em*, *NXspindispersion*

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.

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/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*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXelectronanalyser.nxdl.xml

NXelectrostatic_kicker**Status:**

base class (contribution), extends *NXcomponent*

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*}

voltage set on supply.

read_current: (optional) *NXlog*

current read from supply.

value: (optional) *NX_CHAR* {units=*NX_CURRENT*}

read_voltage: (optional) *NXlog*

voltage read from supply.

value: (optional) *NX_CHAR* {units=*NX_VOLTAGE*}

Hypertext Anchors

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

- */NXelectrostatic_kicker/beamline_distance-field*
- */NXelectrostatic_kicker/description-field*
- */NXelectrostatic_kicker/read_current-group*
- */NXelectrostatic_kicker/read_current/value-field*
- */NXelectrostatic_kicker/read_voltage-group*
- */NXelectrostatic_kicker/read_voltage/value-field*
- */NXelectrostatic_kicker/set_current-field*
- */NXelectrostatic_kicker/set_voltage-field*
- */NXelectrostatic_kicker/timing-field*
- */NXelectrostatic_kicker/timing@description-attribute*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXelectrostatic_kicker.nxdl.xml

NXellipsometry

Status:

application definition (contribution), 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](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

NXenergydispersion**Status:**

base class (contribution), extends [NXcomponent](#)

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

NXfiber**Status:**

base class (contribution), extends [NXcomponent](#)

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 (contribution), 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 (contribution), 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 (contribution), 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

NXisocontour

Status:

base class (contribution), 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 (contribution), 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 (contribution), 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 (contribution), 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_opt

Status:

base class (contribution), extends *NXcomponent*

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 (contribution), extends [NXcomponent](#)

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

NXmatch_filter**Status:**

base class (contribution), 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 (contribution), 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 col-
umn. 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 analyz-
ers. 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 NXener-
gydispersion.

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 a 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 (contribution), 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](#), [NXentry](#), [NXms_feature_set](#), [NXms_snapshot_set](#), [NXms_snapshot](#), [NXobject](#), [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. At 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) *NXobject*

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.

bcd: (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 (contribution), 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 (contribution), 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 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 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: [jl])
{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 (contribution), 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](#), [NXentry](#), [NXfabrication](#), [NXms_snapshot_set](#), [NXms_snapshot](#), [NXobject](#), [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. At 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) *NXObject*

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/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/max_physical_capacity-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/name-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_IO_SYS/CS_IO_OBJ/technology-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_MM_SYS-group*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_MM_SYS/total_physical_memory-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT-group*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/description-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/elapsed_time-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/end_time-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_resident_memory_snapshot-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/max_virtual_memory_snapshot-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_gpus-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_processes-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/number_of_threads-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/CS_PROFILING_EVENT/start_time-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/name-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/operating_system-field*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/operating_system@version-attribute*
- */NXms_score_results/ENTRY/performance/CS_COMPUTER/uuid-field*
- */NXms_score_results/ENTRY/performance/current_working_directory-field*
- */NXms_score_results/ENTRY/performance/end_time-field*
- */NXms_score_results/ENTRY/performance/number_of_gpus-field*
- */NXms_score_results/ENTRY/performance/number_of_processes-field*
- */NXms_score_results/ENTRY/performance/number_of_threads-field*
- */NXms_score_results/ENTRY/performance/start_time-field*
- */NXms_score_results/ENTRY/performance/total_elapsed_time-field*
- */NXms_score_results/ENTRY/PROGRAM-group*

- `/NXms_score_results/ENTRY/PROGRAM/program_name-field`
- `/NXms_score_results/ENTRY/PROGRAM/program_name@version-attribute`
- `/NXms_score_results/ENTRY/results_path-field`
- `/NXms_score_results/ENTRY/ROI_SET-group`
- `/NXms_score_results/ENTRY/ROI_SET/boundary-group`
- `/NXms_score_results/ENTRY/ROI_SET/boundary/boundaries-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 (contribution), 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 (contribution), 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* (contribution), 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 calibration 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]
- pressure: [p1,p1,p1,p2,p2,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, describe 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 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 ‘SPEC-TRUM’ 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

NXorientation_set

Status:

base class (contribution), 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

NXpolarizer_opt

Status:

base class (contribution), extends *NXcomponent*

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-Foucault 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 outgoing (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 the device 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

NXpulser_apm**Status:**

base class (contribution), extends [NXcomponent](#)

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.

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/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

NXquadric

Status:

base class (contribution), 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 (contribution), extends *NXcomponent*

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 (contribution), extends *NXcomponent*

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:

NXtransformations

Structure:

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/TRANSFORMATIONS-group*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXreflectron.nxdl.xml

NXregion

Status:

base class (contribution), extends *NXObject*

Description:

Geometry and logical description of a region of data in a parent group. When used, it could be a child group to, say, *NXdetector*.

This can be used to describe a subset of data used to create downsampled data or to derive some data from that subset.

Note, the fields for the rectangular region specifiers follow HDF5's dataspace hyperslab parameters (see https://portal.hdfgroup.org/display/HDF5/H5S_SELECT_HYPERSLAB). Note when **block** = 1, then **stride** ≡ **step** in Python slicing.

For example, a ROI sum of an area starting at index of [20,50] and shape [220,120] in image data:

```
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 down-sampled 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,

$$(D_0, \dots, D_{\mathbf{O}-1}, d_0, \dots, d_{\mathbf{R}-1})$$

could be split into a left set of **O** outer dimensions, **D**, and a right set of **R** region dimensions, **d**, where the data field has rank **O + R**. Note **O >= 0**.

O: Outer rank

R: Region rank

Groups cited:

NXdata

Structure:

@region_type: (required) *NX_CHAR*

This is `rectangular` to describe the region as a hyper-rectangle in the index space of its parent group's data field.

Obligatory value: `rectangular`

parent: (optional) *NX_CHAR*

The name of data field in the parent group or the path of a data field relative to the parent group (so it could be a field in a subgroup of the parent group)

parent_mask: (optional) *NX_CHAR*

The name of an optional mask field in the parent group with rank **R** and dimensions **d**. For example, this could be `pixel_mask` of an *NXdetector*.

start: (recommended) *NX_NUMBER* (Rank: 1, Dimensions: [R])

The starting position for region in detector data field array. This is recommended as it also defines the region rank. If omitted then defined as an array of zeros.

count: (recommended) *NX_INT* (Rank: 1, Dimensions: [R])

The number of blocks or items in the hyperslab selection. If omitted then defined as an array of dimensions that take into account the other hyperslab selection fields to span the parent data field's shape.

stride: (optional) *NX_INT* (Rank: 1, Dimensions: [R])

An optional field to define striding used to downsample data. If omitted then defined as an array of ones.

block: (optional) *NX_INT* (Rank: 1, Dimensions: [R])

An optional field to define the block size used to copy or downsample data. In the i -th dimension, if $\text{block}[i] < \text{stride}[i]$ then the downsampling blocks have gaps between them; when **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 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 $(D_0, \dots, D_{O-1}, \text{block}[0] * \text{count}[0], \dots, \text{block}[R-1] * \text{count}[R-1])$, otherwise the expected shape is $(D_0, \dots, D_{O-1}, \text{count}[0], \dots, \text{count}[R-1])$.

The following figure shows how blocks are used in downsampling:

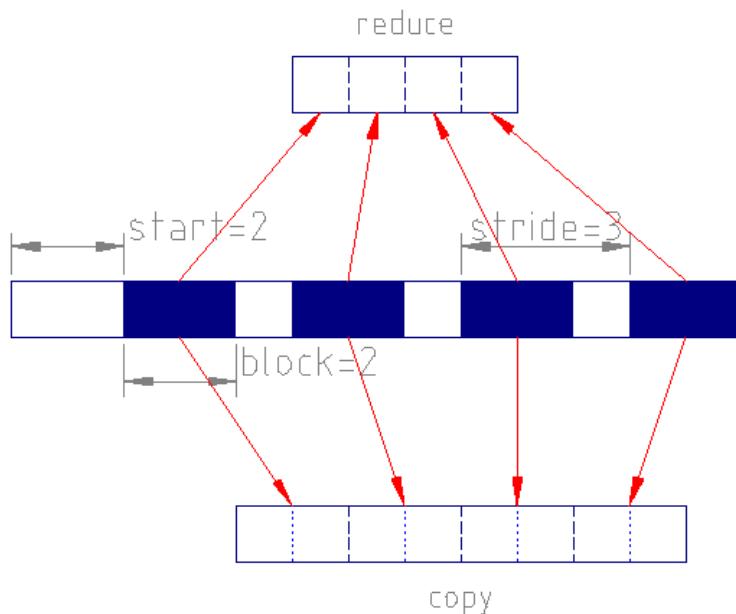


Fig. 14: A selection with **start** = 2, **count** = 4, **stride** = 3, **block** = 2 from a dataset with shape [13] will result in the **reduce** dataset of shape [4] and a **copy** dataset of shape [8].

statistics: (optional) *NXdata*

An optional group containing any statistics derived from the region in parent group's data such as sum, minimum, maximum, mean, mode, median, rms, variance, etc. Where more than one statistical dataset is written (specified with @signal) then add @auxiliary_signals listing the others. All data fields should have shapes of D .

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

NXsensor_scan

Status:

application definition (contribution), 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_controller](#), [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, separate 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: (recommended) *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_CONTROLLER: (required) *NXpid_controller*

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_CONTROLLER-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 (contribution), extends [NXcomponent](#)

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 (contribution), 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 (contribution), 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 (contribution), 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/DASlogs/POSITIONER/average_value_error-field*
- */NXsnsevent/ENTRY/DASlogs/POSITIONER/average_value_errors-field*
- */NXsnsevent/ENTRY/DASlogs/POSITIONER/description-field*
- */NXsnsevent/ENTRY/DASlogs/POSITIONER/duration-field*
- */NXsnsevent/ENTRY/DASlogs/POSITIONER/maximum_value-field*
- */NXsnsevent/ENTRY/DASlogs/POSITIONER/minimum_value-field*
- */NXsnsevent/ENTRY/DASlogs/POSITIONER/time-field*
- */NXsnsevent/ENTRY/DASlogs/POSITIONER/value-field*
- */NXsnsevent/ENTRY/DATA-group*
- */NXsnsevent/ENTRY/DATA/data_x_y-link*

- */NXsnsevent/ENTRY/DATA/x_pixel_offset-link*
- */NXsnsevent/ENTRY/DATA/y_pixel_offset-link*
- */NXsnsevent/ENTRY/definition-field*
- */NXsnsevent/ENTRY/duration-field*
- */NXsnsevent/ENTRY/end_time-field*
- */NXsnsevent/ENTRY/entry_identifier-field*
- */NXsnsevent/ENTRY/EVENT_DATA-group*
- */NXsnsevent/ENTRY/EVENT_DATA/event_index-link*
- */NXsnsevent/ENTRY/EVENT_DATA/event_pixel_id-link*
- */NXsnsevent/ENTRY/EVENT_DATA/event_time_of_flight-link*
- */NXsnsevent/ENTRY/EVENT_DATA/pulse_time-link*
- */NXsnsevent/ENTRY/experiment_identifier-field*
- */NXsnsevent/ENTRY/instrument-group*
- */NXsnsevent/ENTRY/instrument/APERTURE-group*
- */NXsnsevent/ENTRY/instrument/APERTURE/origin-group*
- */NXsnsevent/ENTRY/instrument/APERTURE/origin/orientation-group*
- */NXsnsevent/ENTRY/instrument/APERTURE/origin/orientation/value-field*
- */NXsnsevent/ENTRY/instrument/APERTURE/origin/shape-group*
- */NXsnsevent/ENTRY/instrument/APERTURE/origin/shape/description-field*
- */NXsnsevent/ENTRY/instrument/APERTURE/origin/shape/shape-field*
- */NXsnsevent/ENTRY/instrument/APERTURE/origin/shape/size-field*
- */NXsnsevent/ENTRY/instrument/APERTURE/origin/translation-group*
- */NXsnsevent/ENTRY/instrument/APERTURE/origin/translation/distance-field*
- */NXsnsevent/ENTRY/instrument/APERTURE/x_pixel_offset-field*
- */NXsnsevent/ENTRY/instrument/ATTENUATOR-group*
- */NXsnsevent/ENTRY/instrument/ATTENUATOR/distance-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/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 (contribution), 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/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 (contribution), extends [NXcomponent](#)

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 (contribution), extends *NXObject*

Description:

the head node for constructively defined geometry

Symbols:

No symbol table

Groups cited:

NXcsg, *NXoff_geometry*, *NXquadric*

Structure:

QUADRATIC: (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/QUADRATIC-group*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsolid_geometry.nxdl.xml

NXspatial_filter

Status:

base class (contribution), 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

NXspin_rotator

Status:

base class (contribution), extends [NXcomponent](#)

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 (contribution), extends [NXcomponent](#)

Description:

Subclass of NXelectronanalyser to describe the spin filters in photoemission experiments.

Symbols:

No symbol table

Groups cited:

[NXdeflector](#), [NXlens_em](#)

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 Sherman 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

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/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/type-field](#)

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspindispersion.nxdl.xml

NXsubsampling_filter

Status:

base class (contribution), 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

NXsubstance

Status:

base class (contribution), extends [NXobject](#)

Description:

A form of matter with a constant, definite chemical composition.

Examples can be single chemical elements, chemical compounds, or alloys. For further information, see https://en.wikipedia.org/wiki/Chemical_substance.

Symbols:

No symbol table

Groups cited:

[NXnote](#)

Structure:

name: (optional) [NX_CHAR](#)

User-defined chemical name of the substance

molecular_mass: (optional) [NX_FLOAT](#) {units=[NX_MOLECULAR_WEIGHT](#)}

Molecular mass of the substance

molecular_formula_hill: (optional) [NX_CHAR](#)

The chemical formula specified using CIF conventions. Abbreviated version of CIF standard:107 This is the *Hill* system used by Chemical Abstracts.

- 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.

identifier_cas: (optional) [NX_CHAR](#)

Unique CAS REGISTRY URI. For further information, see <https://www.cas.org/>.

@type: (optional) [NX_CHAR](#)

Obligatory value: URL

@cas_number: (optional) *NX_CHAR*

Numeric CAS REGISTRY number associated with this identifier.

@cas_name: (optional) *NX_CHAR*

CAS REGISTRY name associated with this identifier.

identifier_inchi_str: (optional) *NX_CHAR*

Standard string InChi identifier” (as per v1.02).

The InChI identifier expresses chemical structures in terms of atomic connectivity, tautomeric state, isotopes, stereochemistry and electronic charge in order to produce a string of machine-readable characters unique to the respective molecule. For further information, see <https://iupac.org/who-we-are/divisions/division-details/inchi/>.

identifier_inchi_key: (optional) *NX_CHAR*

Condensed, 27 character InChI key. Hashed version of the full InChI (using the SHA-256 algorithm).

identifier_iupac_name: (optional) *NX_CHAR*

Name according to the IUPAC system (standard). For further information, see <https://iupac.org/>.

identifier_smiles: (optional) *NX_CHAR*

Identifier in the SMILES (Simplified Molecular Input Line Entry System) system For further information, see <https://www.daylight.com/smiles/>.

identifier_canonical_smiles: (optional) *NX_CHAR*

Canonical version of the SMILES identifier

identifier_pub_chem: (optional) *NX_CHAR*

Standard PubChem identifier (CID).

The PubChem Compound Identifier (CID) is a unique numerical identifier assigned to a compound in the PubChem database, which contains information on the biological activities of small molecules. The CID allows users to access detailed data about compounds, including their chemical structure, molecular formula, and biological properties.

For further information, see <https://pubchem.ncbi.nlm.nih.gov/>.

@pub_chem_link: (optional) *NX_CHAR*

CAS REGISTRY name associated with this identifier.

cas_image: (optional) *NXnote*

CAS REGISTRY image

Hypertext Anchors

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

- /NXsubstance/cas_image-group
- /NXsubstance/identifier_canonical_smiles-field
- /NXsubstance/identifier_cas-field

- */NXsubstance/identifier_cas@cas_name-attribute*
- */NXsubstance/identifier_cas@cas_number-attribute*
- */NXsubstance/identifier_cas@type-attribute*
- */NXsubstance/identifier_inchi_key-field*
- */NXsubstance/identifier_inchi_str-field*
- */NXsubstance/identifier_iupac_name-field*
- */NXsubstance/identifier_pub_chem-field*
- */NXsubstance/identifier_pub_chem@pub_chem_link-attribute*
- */NXsubstance/identifier_smiles-field*
- */NXsubstance/molecular_formula_hill-field*
- */NXsubstance/molecular_mass-field*
- */NXsubstance/name-field*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsubstance.nxdl.xml

NXtransmission**Status:**

application definition (contribution), 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 research 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_wavelengths

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 (contribution), extends [NXcomponent](#)

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_refraction_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_refraction_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 (contribution), 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

$$g_2(\mathbf{Q}, t) = \frac{\langle I(\mathbf{Q}, t)I(\mathbf{Q}, t + t) \rangle}{\langle I(\mathbf{Q}, t) \rangle^2}; t > 0$$

Typically, g_2 is a quantity calculated for a group of pixels representing a specific region of reciprocal space. These groupings, or bins, are generically described as q . Some open-source XPCS libraries refer to these bins as “rois”, which are not to be confused with EPICS AreaDetector ROI. See usage guidelines for `q_lists` and `roi_maps` within a mask.¹

In short, g_2 should be ordered according to the `roi_map` value. In principle, any format is acceptable if the data and its axes are self-describing as per NeXus recommendations. However, the data is preferred in one of the following two formats:

- iterable list of linked files (or keys) for each g_2 with 1 file (key) per q , where q is called by the nth `roi_map` value
- 2D array² with shape (g_2, q) , where q is represented by the nth `roi_map` value, not the value q value

¹ mask: `NXxpcss:/entry/instrument/masks-group`

² NeXus 2-D data and axes: https://manual.nexusformat.org/classes/base_classes/NXdata.html#nxdata

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 *g₂* 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*}

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:

$$C(\mathbf{Q}, t_1, t_2) = \frac{\langle I(\mathbf{Q}, t_1)I(\mathbf{Q}, t_2) \rangle}{\langle I(\mathbf{Q}, t_1) \rangle \langle I(\mathbf{Q}, t_2) \rangle}$$

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 documentation 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 $(g_2, q, \text{nth_partial})$

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.

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.

³ Coherent X-ray Imaging Data Bank: <https://cxdidb.org/cxi.html>

“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 NXcps(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. Note that the value of 0 represents in-action. XPCS computations are performed on all pixels with a value > 0.

The units attribute should be set to "au" indicating arbitrary units.

dynamic_q_list: (optional) *NX_NUMBER* {units=[NX_PER_LENGTH](#)}

1-D list of Q values, one for each roi index value.

List order is determined by the index value of the associated roi map starting at 1.

The only requirement for the list is that it may be iterable. Some expected formats are:

- iterable list of floats (i.e., $Q(r)$)
- iterable list of tuples (i.e., $Q(r), \varphi$), but preferable use the 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.

- */NXpcs/ENTRY-group*
- */NXpcs/ENTRY/data-group*
- */NXpcs/ENTRY/data/delay_difference-field*
- */NXpcs/ENTRY/data/delay_difference@storage_mode-attribute*
- */NXpcs/ENTRY/data/frame_average-field*
- */NXpcs/ENTRY/data/frame_sum-field*
- */NXpcs/ENTRY/data/g2-field*
- */NXpcs/ENTRY/data/g2@storage_mode-attribute*
- */NXpcs/ENTRY/data/g2_derr-field*
- */NXpcs/ENTRY/data/g2_derr@storage_mode-attribute*
- */NXpcs/ENTRY/data/G2_unnormalized-field*
- */NXpcs/ENTRY/data/G2_unnormalized@storage_mode-attribute*
- */NXpcs/ENTRY/definition-field*
- */NXpcs/ENTRY/end_time-field*
- */NXpcs/ENTRY/entry_identifier-field*
- */NXpcs/ENTRY/entry_identifier_uuid-field*
- */NXpcs/ENTRY/instrument-group*

- */NXpcss/ENTRY/instrument/DETECTOR-group*
- */NXpcss/ENTRY/instrument/DETECTOR/beam_center_x-field*
- */NXpcss/ENTRY/instrument/DETECTOR/beam_center_y-field*
- */NXpcss/ENTRY/instrument/DETECTOR/count_time-field*
- */NXpcss/ENTRY/instrument/DETECTOR/description-field*
- */NXpcss/ENTRY/instrument/DETECTOR/distance-field*
- */NXpcss/ENTRY/instrument/DETECTOR/frame_time-field*
- */NXpcss/ENTRY/instrument/DETECTOR/x_pixel_size-field*
- */NXpcss/ENTRY/instrument/DETECTOR/y_pixel_size-field*
- */NXpcss/ENTRY/instrument/incident_beam-group*
- */NXpcss/ENTRY/instrument/incident_beam/extents-field*
- */NXpcss/ENTRY/instrument/incident_beam/incident_energy-field*
- */NXpcss/ENTRY/instrument/incident_beam/incident_energy_spread-field*
- */NXpcss/ENTRY/instrument/incident_beam/incident_polarization_type-field*
- */NXpcss/ENTRY/instrument/masks-group*
- */NXpcss/ENTRY/instrument/masks/dynamic_phi_list-field*
- */NXpcss/ENTRY/instrument/masks/dynamic_q_list-field*
- */NXpcss/ENTRY/instrument/masks/dynamic_roi_map-field*
- */NXpcss/ENTRY/instrument/masks/static_q_list-field*
- */NXpcss/ENTRY/instrument/masks/static_roi_map-field*
- */NXpcss/ENTRY/NOTE-group*
- */NXpcss/ENTRY/sample-group*
- */NXpcss/ENTRY/sample/position_x-group*
- */NXpcss/ENTRY/sample/position_y-group*
- */NXpcss/ENTRY/sample/position_z-group*
- */NXpcss/ENTRY/sample/temperature-field*
- */NXpcss/ENTRY/sample/temperature_set-field*
- */NXpcss/ENTRY/scan_number-field*
- */NXpcss/ENTRY/start_time-field*
- */NXpcss/ENTRY/twotime-group*
- */NXpcss/ENTRY/twotime/g2_err_from_twotime_corr_func-field*
- */NXpcss/ENTRY/twotime/g2_err_from_twotime_corr_func@storage_mode-attribute*
- */NXpcss/ENTRY/twotime/g2_err_from_twotime_corr_func_partials-field*
- */NXpcss/ENTRY/twotime/g2_from_twotime_corr_func-field*
- */NXpcss/ENTRY/twotime/g2_from_twotime_corr_func@baseline_reference-attribute*
- */NXpcss/ENTRY/twotime/g2_from_twotime_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
<code>/_static/nxdl_vocabulary.html</code>	Human-readable HTML list of anchors, by vocabulary term, with links to the manual.
<code>/_static/nxdl_vocabulary.json</code>	vocabulary list by key in JSON format ²
<code>/_static/nxdl_vocabulary.txt</code>	list of all anchors, sorted alphabetically
<code>/_static/nxdl_vocabulary.yml</code>	vocabulary list by key in YAML format ³

² JSON: <https://www.w3schools.com/json/>

³ YAML <https://yaml.org>

NAPI: NEXUS APPLICATION PROGRAMMER INTERFACE (FROZEN)

4.1 Status

This application program interface (API) was developed to support the reading and writing of NeXus files through unified function calls, regardless of the physical data format (XML, HDF4, HDF5).

In the meantime it has been decided that active development of NeXus definitions and tools will concentrate on HDF5 as the only supported physical data format. It is expected that most application developers will use standard HDF5 tools to read and write NeXus. Two examples are provided in [HDF5 in C with libhdf5](#).

Therefore, the decision has been taken to freeze the NAPI. Maintenance is reduced to bug fixes.

4.2 Overview

The core routines have been written in C but wrappers are available for a number of other languages including C++, Fortran 77, Fortran 90, Java, Python and IDL. The API makes the reading and writing of NeXus files transparent; the user doesn't even need to know the underlying format when reading a file since the API calls are the same.

The NeXus Application Programming Interface for the various language backends is available on-line from <https://github.com/nexusformat/code/>

The NeXusIntern.pdf document (<https://github.com/nexusformat/code/blob/master/doc/api/NeXusIntern.pdf>) describes the internal workings of the NeXus-API. You are very welcome to read it, but it will not be of much use if all you want is to read and write files using the NAPI.

The NeXus Application Program Interface call routines in the appropriate backend (HDF4, HDF5 or XML) to read and write files with the correct structure. The API serves a number of purposes:

1. It simplifies the reading and writing of NeXus files.
2. It ensures a certain degree of compliance with the NeXus standard.
3. It hides the implementation details of the format. In particular, the API can read and write HDF4, HDF5, and XML files using the same routines.

4.3 Core API

The core API provides the basic routines for reading, writing and navigating NeXus files. Operations are performed using a handle that keeps a record of its current position in the file hierarchy. All 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`

```
1 #!/sbin/sh
2 java -classpath /usr/lib/classes.zip:../jnexus.jar: \
3     -Dorg.nexusformat.JNEXUSLIB=../libjnexus.so TestJapi
```

Windows 32 example batch file to start `jnexus.jar`

```
1 set JL=-Dorg.nexusformat.JNEXUSLIB=..\jnexus\bin\win32\jnexus.dll
2 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

```

1  try{
2      NexusFile nf = new NexusFile(filename, NexusFile.NXACC_READ);
3      nf.opengroup("entry1","NXentry");
4      nf.finalize();
5  }catch(NexusException ne) {
6      // Something was wrong!
7 }
```

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

```

1  int idata[][] = new idata[10][20];
2  int iDim[] = new int[2];
3
4  // put some data into idata.....
5
6  // write idata
7  iDim[0] = 10;
8  iDim[1] = 20;
9  nf.makedata("idata",NexusFile.NX_INT32,2,iDim);
10 nf.opendata("idata");
11 nf.putdata(idata);
12
13 // read idata
14 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

```
1  String ame = "Alle meine Entchen";
2  nf.makedata("string_data",NexusFile.NX_CHAR,
3      1,ame.length()+2);
4  nf.opendata("string_data");
5  nf.putdata(ame.getBytes());
```

And reading:

String reading

```
1  byte bData[] = new byte[132];
2  nf.opendata("string_data");
3  nf.getdata(bData);
4  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()

```
1  /* C -API */
2  NXstatus NXgetinfo(NXhandle handle, int *rank, int iDim[],
3                      int *datatype);
```

Java API signature of getinfo()

```
1  // Java
2  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

```

1 nf.opengroup(group,nxclass);
2 h = nf.groupdir();
3 e = h.keys();
4 System.out.println("Found in vGroup entry:");
5 while(e.hasMoreElements())
6 {
7     vname = (String)e.nextElement();
8     vclass = (String)h.get(vname);
9     System.out.println("      Item: " + vname + " class: " + vclass);
10 }

```

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

```

1 Hashtable h = nf.attrdir();
2 Enumeration e = h.keys();
3 while(e.hasMoreElements())
4 {
5     attname = (String)e.nextElement();
6     atten = (AttributeEntry)h.get(attname);
7     System.out.println("Found global attribute: " + attname +
8         " type: " + atten.type + " ,length: " + atten.length);
9 }

```

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>)

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. 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.

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

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>

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 your 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 removed (so --buildroot / is a really bad idea). Only change buildroot 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 wiki¹ of the NeXus definitions repository². 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>

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 files¹ which you may browse and download. There is a cursory analysis² of every file in this repository as to whether it can be read as HDF5 or NeXus HDF5. The analysis code³, 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
```

nx ingest

nx ingest 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.

nxssummary

Use **nxssummary** 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/exmpledata>

² <https://github.com/nexusformat/exmpledata/blob/master/critique.md>

³ <https://github.com/nexusformat/exmpledata/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/exampledadata/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 analysis. 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:](https://)

[//archive.ansto.gov.au/ResearchHub/OurInfrastructure/ACNS/Facilities/Computing/GumTree/index.htm](http://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://nepy.github.io/nepy/>)

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 SPEC⁴ 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.

HDFFexplorer (<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 identified⁵ 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 IgorPro¹⁰ 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 read¹² and write¹³ 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.

7.6.5 Language API: Python

- **Dials**¹⁵ has python (and some C++) code for reading `NXmx`¹⁶
 - `cctbx.xfel` code for writing¹⁷ `NXmx` master files for JF16M at SwissFEL
- **h5py**¹⁸
HDF5 for Python (h5py) is a general-purpose Python interface to HDF5.
- **Mantis**¹⁹, with NeXus-specific python code²⁰, reads `NXstxm`

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/prjeman/hdf5gateway>

¹⁰ IgorPro: <https://wavemetrics.com>

¹¹ <https://dawnscl.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/dawnscl/blob/master/org.eclipse.dawnscl.hdf5/src/org/eclipse/dawnscl/hdf5/nexus/NexusFileHDF5.java>

¹⁴ <https://github.com/nexusformat/definitions/files/4107439/NXreader.zip>

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/>

- **nexusformat**²¹ NeXus package for Python
Provides an API to open, create, plot, and manipulate NeXus data.
- **SasView**²² has python code to read²³ and write²⁴ *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

²¹ <https://github.com/nexpy/nexusformat>

²² <https://www.sasview.org/>

²³ read: https://github.com/SasView/blob/master/src/sas/sascalc/dataloader/readers/cansas_reader_HDF5.py

²⁴ write: https://github.com/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 from 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

Könnecke (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 format² as a standard for data storage at APS

1994-06:

Mark Könnecke (then ISIS, now PSI) made a proposal using netCDF³ for the European neutron scattering community while working at ISIS

² https://www.nexusformat.org/pdfs/Proposed_Data_Standard_for_theAPS.pdf

³ <https://www.nexusformat.org/pdfs/European-Formats.pdf>

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

Published by NeXus International Advisory Committee, <https://www.nexusformat.org>

Copyright (C) 1996-2024 NeXus International Advisory Committee (NIAC)

The NeXus manual is licensed under the terms of the GNU Free Documentation License version 1.3.

download

FDL

GNU

<http://www.gnu.org/licenses/fdl-1.3.txt>

The code examples in the NeXus manual are licensed under the terms of the GNU Lesser General Public License version 3.

download

LGPL

GNU

<http://www.gnu.org/licenses/lgpl-3.0.txt>

Publishing Information

This manual built May 21, 2025.

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

a (*field*), 416, 482
a_b_c (*field*), 416
Abbe_number (*field*), 938
aberration (*base class*), *see* NXaberration, 166
abrasive_medium (*field*), 934
abrasive_medium_carrier (*field*), 934
absorbing_material (*field*), 194, 288
absorption_cross_section (*field*), 174
acceleration_time (*field*), 357
acceptance_angle (*field*), 926, 998
accepted_photon_beam_divergence (*field*), 183
accepting_aperture (*field*), 188
acquisition_mode (*field*), 239, 429
acquisition_time (*field*), 191
activity (*base class*), *see* NXactivity, 168
actuation_target (*field*), 169, 327–329
actuator (*base class*), *see* NXactuator, 168
additional_phase_information (*field*), 909
address (*field*), 418, 456, 603, 631, 712, 721, 729, 738, 777, 786, 793, 801, 812, 825, 942, 950, 970, 984, 1010, 1044
address, absolute, 20
address, relative, 20
aequatorial_angle (*field*), 536, 537, 540
affiliation (*field*), 418, 456, 603, 631, 712, 721, 729, 738, 777, 786, 793, 801, 812, 825, 942, 950, 970, 983, 1010, 1044
algorithm (*field*), 340, 455, 469, 473, 481, 482
alias (*field*), 167, 319, 459–461, 463, 481, 606
alpha (*field*), 416, 482, 579, 645, 802, 846
alpha_beta_gamma (*field*), 416
alpha_value_choice (*field*), 700
alpha_values (*field*), 701
alternative (*field*), 986
amplifier_bias (*field*), 260
amplifier_type (*field*), 260, 944
amplifier_voltage (*field*), 260
aN (*field*), 186
analysis_description (*field*), 641, 649, 654, 661, 680, 687, 690, 698, 704, 709, 712, 721, 729, 737, 777, 786, 792, 800, 811, 824, 961, 969

analysis_identifier (*field*), 640, 649, 654, 661, 680, 687, 690, 698, 704, 709, 711, 721, 729, 737, 777, 786, 792, 800, 811, 824, 961, 969
analyze_coprecipitation (*field*), 655
analyze_intersection (*field*), 655
analyze_proximity (*field*), 655
angle (*field*), 167
angle_of_incidence (*field*), 984
angles (*field*), 296, 429
angular_calibration (*field*), 239, 520
angular_calibration_applied (*field*), 239, 520
angular_dispersion (*field*), 837, 838, 1045
angular_resolution (*field*), 915
angular_spread (*field*), 921
anode_material (*field*), 397
AOI_range (*field*), 842
aperture (*base class*), *see* NXaperture, 170
aperture (*field*), 421
API, *see* NAPI
 F77; POLDI, 1085
 IDL; aXis2000, 1085
 IgorPro; HDF5gateway, 1085
 java; Dawn, 1085
 java; NXreader.zip, 1085
 mixed; FOCUS, 1086
 mixed; ZEBRA, 1086
 Python; Dials, 1085
 Python; h5py, 1085
 Python; Mantis, 1085
 Python; nexusformat, 1085
 Python; SasView, 1086
apm (*application definition*), *see* NXapm, 599
apm_composition_space_results
 (*application definition*), *see*
 NXapm_composition_space_results, 629
apm_input_ranging (*base class*), *see*
 NXapm_input_ranging, 638
apm_input_reconstruction (*base class*), *see*
 NXapm_input_reconstruction, 639
apm_paraprobe_config_clusterer
 (*application definition*), *see*
 NXapm_paraprobe_config_clusterer, 640

apm_paraprobe_config_distancer (application definition), see NXapm_paraprobe_config_distancer, 648		apm_paraprobe_results_transcoder (application definition), see NXapm_paraprobe_results_transcoder, 824
apm_paraprobe_config_intersector (application definition), see NXapm_paraprobe_config_intersector, 654		ApmAppDef, 594
apm_paraprobe_config_nanochem (application definition), see NXapm_paraprobe_config_nanochem, 660		ApmBC, 594
apm_paraprobe_config_ranger (application definition), see NXapm_paraprobe_config_ranger, 679		application definition, 422
apm_paraprobe_config_selector (application definition), see NXapm_paraprobe_config_selector, 686		applied (field), 185, 195, 201, 255, 258, 370, 476, 478, 479, 609, 945, 946
apm_paraprobe_config_spatstat (application definition), see NXapm_paraprobe_config_spatstat, 689		apply (field), 965
apm_paraprobe_config_surfacer (application definition), see NXapm_paraprobe_config_surfacer, 697		archive (application definition), see NXarchive, 424
apm_paraprobe_config_tessellator (application definition), see NXapm_paraprobe_config_tessellator, 704		area (field), 417, 747, 756, 758, 867, 878
apm_paraprobe_config_transcoder (application definition), see NXapm_paraprobe_config_transcoder, 709		arpes (application definition), see NXarpes, 428
apm_paraprobe_results_clusterer (application definition), see NXapm_paraprobe_results_clusterer, 711		arrival_time_pairs (field), 614
apm_paraprobe_results_distanter (application definition), see NXapm_paraprobe_results_distanter, 720		aspect (field), 750
apm_paraprobe_results_intersector (application definition), see NXapm_paraprobe_results_intersector, 728		assumed_composition_isotopes (field), 682
apm_paraprobe_results_nanochem (application definition), see NXapm_paraprobe_results_nanochem, 736		atom (base class), see NXatom, 172
apm_paraprobe_results_ranger (application definition), see NXapm_paraprobe_results_ranger, 776		atom_types (field), 271, 458, 483, 606, 909, 946, 990
apm_paraprobe_results_selector (application definition), see NXapm_paraprobe_results_selector, 785		attached_to (field), 389
apm_paraprobe_results_spatstat (application definition), see NXapm_paraprobe_results_spatstat, 791		attenuation (field), 837, 926
apm_paraprobe_results_surfacer (application definition), see NXapm_paraprobe_results_surfacer, 799		attenuator (base class), see NXattenuator, 174
apm_paraprobe_results_tessellator (application definition), see NXapm_paraprobe_results_tessellator, 810		attenuator_transmission (field), 174, 518, 584, 837, 1045
		attribute, see field attribute, see group attribute, see file attribute, 133, see NXDL attribute
		HDF, 55
		NXclass, 38
		attribute (NXDL element), 135
		attributeType (NXDL data type), 139
		author (field), 340, 1018, 1027
		authors, 1091
		automatic plotting, see plotting
		auxiliary_signals (file attribute), 229
		average_power (field), 178
		average_value (field), 326, 1018, 1027
		average_value_error (field), 326, 1018, 1027
		average_value_errors (field), 326, 1018, 1027
		axes (attribute), 53, 227
		axes (field attribute), 231, 898
		axes (file attribute), 230
		axes (group attribute), 243, 300, 469–475, 482, 483, 618, 619, 744, 745, 898, 899, 992, 993, 1047
		axis, 54
		axis (field attribute), 231, 235, 236
		axis_angle_convention (field), 459, 889, 951, 971
		axis_energy (field), 271, 401–405, 473–475, 483
		axis_i (field), 306–310, 401, 402, 404, 405, 470–475, 483
		axis_j (field), 307, 309, 310, 402, 404, 405, 470–475, 483
		axis_k (field), 307, 310, 402, 404, 470, 472, 474, 475
		axis_mass_to_charge (field), 619
		axis_x (field), 269, 483, 618
		axis_y (field), 269, 482, 618
		axis_z (field), 482, 618

A

AXISNAME (*field*), 230, 413, 834
 AXISNAME_end (*field*), 414
 AXISNAME_increment_set (*field*), 414
 AXISNAME_indices (*file attribute*), 230
 AXISNAME_indices (*group attribute*), 469–475, 482, 483, 618, 619
 azimuthal (*field*), 544
 azimuthal_angle (*field*), 207, 237, 367, 511, 536, 540, 555, 558, 561, 1019, 1028
 azimuthal_width (*field*), 544

B

b (*field*), 416, 482
 background_mean (*field*), 366
 backside_roughness (*field*), 922
 bandwidth (*field*), 190, 313, 836
 base_class, 161
 base_temperature (*field*), 610
 baseline_reference (*field attribute*), 1054–1056
 basicComponent (*NXDL data type*), 154
 beam (*base class*), *see* NXbeam, 175
 beam_center_derived (*field*), 519
 beam_center_x (*field*), 239, 442, 519, 536, 540, 584, 899, 1057
 beam_center_y (*field*), 239, 443, 520, 536, 540, 584, 899, 1057
 beam_path (*base class*), *see* NXbeam_path, 834
 beam_position (*field*), 253
 beam_profile (*field*), 836
 beam_shape (*field*), 444
 beam_size_x (*field*), 444
 beam_size_y (*field*), 444
 beam_splitter (*base class*), *see* NXbeam_splitter, 841
 beam_stop (*base class*), *see* NXbeam_stop, 181
 BEAMdirection (*field*), 179
 beamline (*field*), 1019, 1028
 beamline_distance (*field*), 917, 940, 1005, 1013, 1035, 1038
 bend_angle_x (*field*), 299, 331
 bend_angle_y (*field*), 299, 331
 bending_magnet (*base class*), *see* NXbending_magnet, 182
 bending_radius (*field*), 183
 beta (*field*), 416, 482
 bibtex (*field*), 193
 binary_data, 45, *see* NX_BINARY
 binary_executable, *see* NAPI installation
 bit_depth_readout (*field*), 241, 521
 bitdepth (*field*), 643, 650, 662, 681, 693, 700, 706, 713, 722, 723, 739, 779, 787, 794, 802, 803, 813–816, 892
 blade_spacing (*field*), 194
 blade_thickness (*field*), 194
 blaze_wavelength (*field*), 838, 1046

C

c
 code_examples, 77
 c (*field*), 416, 482
 calibrated_axis (*field*), 186, 945, 946
 calibrated_tof (*field*), 616
 calibration (*base class*), *see* NXcalibration, 184
 calibration_data_link (*field*), 985
 calibration_date (*field*), 238
 calibration_status (*field*), 984
 calibration_time (*field*), 985, 1011
 camera_length (*field*), 345
 candidates (*field*), 673
 canSAS, 431
 canSAS (*application definition*), *see* NXcanSAS, 431
 canSAS_class (*group attribute*), 433, 434, 441–446
 cap_surface_area (*field*), 848
 capabilities (*field*), 608–612, 633, 634, 716, 724, 732, 733, 760, 761, 781, 788, 795, 796, 805, 806, 818, 829, 830, 975, 976
 capability (*field*), 287
 capillary (*base class*), *see* NXcapillary, 187
 cardinality (*field*), 631, 643, 649, 650, 661, 662, 681, 687, 688, 692, 693, 699, 705, 714, 741, 746, 756–759, 816, 848, 849, 855, 860, 864, 866, 867, 869, 871, 874, 876, 878, 880, 929, 958, 971, 1014
 cas_name (*field attribute*), 1042
 cas_number (*field attribute*), 1042
 categorical_label (*field*), 882, 1015
 category (*NXDL attribute*), 65
 cdeform_field (*field*), 255
 cell_dimensions (*field*), 632, 741, 855, 972
 cell_size (*field*), 961
 center (*field*), 643, 649, 650, 661, 662, 681, 687, 692, 699, 705, 748, 750, 759, 848, 850, 860, 864, 868, 869, 874, 876, 878
 center_energy (*field*), 924
 central_stop_diameter (*field*), 294
 central_stop_material (*field*), 294
 central_stop_thickness (*field*), 294
 cg_alpha_complex (*base class*), *see* NXcg_alpha_complex, 845

cg_cylinder_set (*base class*), *see* NXcg_cylinder_set, 847
 cg_ellipsoid_set (*base class*), *see* NXcg_ellipsoid_set, 849
 cg_face_list_data_structure (*base class*), *see* NXcg_face_list_data_structure, 850
 cg_geodesic_mesh (*base class*), *see* NXcg_geodesic_mesh, 853
 cg_grid (*base class*), *see* NXcg_grid, 854
 cg_half_edge_data_structure (*base class*), *see* NXcg_half_edge_data_structure, 856
 cg_hexahedron_set (*base class*), *see* NXcg_hexahedron_set, 858
 cg_marching_cubes (*base class*), *see* NXcg_marching_cubes, 862
 cg_parallelogram_set (*base class*), *see* NXcg_parallelogram_set, 863
 cg_point_set (*base class*), *see* NXcg_point_set, 865
 cg_polygon_set (*base class*), *see* NXcg_polygon_set, 866
 cg_polyhedron_set (*base class*), *see* NXcg_polyhedron_set, 869
 cg_polyline_set (*base class*), *see* NXcg_polyline_set, 871
 cg_roi_set (*base class*), *see* NXcg_roi_set, 873
 cg_sphere_set (*base class*), *see* NXcg_sphere_set, 874
 cg_tetrahedron_set (*base class*), *see* NXcg_tetrahedron_set, 875
 cg_triangle_set (*base class*), *see* NXcg_triangle_set, 877
 cg_triangulated_surface_mesh (*base class*), *see* NXcg_triangulated_surface_mesh, 879
 cg_unit_normal_set (*base class*), *see* NXcg_unit_normal_set, 880
 CgmsAppDef, 596
 CgmsBC, 596
 chamber (*base class*), *see* NXchamber, 881
 changer_position (*field*), 379, 1021, 1030
 charge (*field*), 172, 751, 752
 charge_state (*field*), 173, 620, 778–780, 828
 charge_vector (*field*), 828
 check_sum (*field attribute*), 236
 checksum (*field*), 340, 455, 469, 473, 481, 482
 chemical_composition (*base class*), *see* NXchemical_composition, 188
 chemical_formula (*field*), 205, 290, 378, 385, 427, 886, 909, 946, 990
 chi (*field*), 578
 chirp_GDD (*field*), 179
 chirp_type (*field*), 179
 choice, 134
 choice (*NXDL element*), 136
 choiceType (*NXDL data type*), 151
 chromatic (*field*), 938
 CIF, 27
 circuit (*base class*), *see* NXcircuit, 190
 cite (*base class*), *see* NXcite, 192
 clad_diameter (*field*), 926
 clad_index_of_refraction (*field*), 927
 clad_material (*field*), 926
 class definitions, 160, *see base class, see application definition, see contributed definition*
 class path, 20
 clear_aperture (*field*), 842, 1049
 cluster_composition (*field*), 731
 cluster_dict_keyword (*field*), 632, 633
 cluster_dict_value (*field*), 632, 633
 cluster_identifier (*field*), 632, 633, 731
 cluster_selection_epsilon (*field*), 645
 cluster_statistics (*field*), 732
 clustering (*base class*), *see* NXclustering, 881
 cnxvalidate (*utility*), 1082
 coating_diameter (*field*), 927
 coating_material (*field*), 291, 297, 299, 331, 844, 927, 939, 1000, 1050
 coating_roughness (*field*), 291, 297, 299, 331
 coating_thickness (*field*), 844, 939, 1000, 1050
 coating_type (*field*), 844, 939, 1000, 1050
 coherence_length (*field*), 836
 collection (*base class*), *see* NXcollection, 193
 collection_description (*field*), 277, 408, 425
 collection_identifier (*field*), 277, 408, 425, 1017, 1026
 collection_time (*field*), 278, 409, 425
 collection_title (*field*), 1017, 1026
 collectioncolumn (*base class*), *see* NXcollectioncolumn, 883
 collimator (*base class*), *see* NXcollimator, 193
 colloquial_name (*field*), 909
 command1 (*field*), 1018, 1027
 command_line_call (*field*), 213, 633, 715, 724, 732, 760, 780, 788, 795, 805, 818, 829, 975
 comment (*field attribute*), 278, 319, 409
 comment (*field*), 981, 982
 common_beam_depolarizer (*field*), 1045
 community, 1073
 comp_current (*field*), 293
 comp_turns (*field*), 293
 company (*field*), 920, 984
 complex (*field*), 306–309
 component (*base class*), *see* NXcomponent, 195
 component (*field*), 381
 component_index (*field*), 296
 components (*field*), 190
 composition (*field*), 189, 356, 605, 632
 composition_error (*field*), 605
 composition_errors (*field*), 189
 composition_product (*field*), 779

computation_method (field), 954
concentration (field), 381
config_filename (field), 630, 712, 721, 729, 737, 777, 786, 792, 800, 811, 825, 969
configuration (field attribute), 278, 409
connections (field), 190
constitution, 74, 1073
construction_date (field), 287
construction_year (field), 920, 984
container (base class), *see* NXcontainer, 884
context (field), 306, 401, 469, 473
contribute, 74
contributed definition, 585, 1073
control_action (field), 354
control_points (field), 672
controller_record (field), 357
convention (field), 904, 907, 910, 911
conversion, 1081
coordinate (field), 632, 855, 973
coordinate systems, 29

- CIF, 29
- IUCr, 25
- McStas, 25, 29, 30
- NeXus, 24
- NeXus polar coordinate, 29
- spherical polar, 30
- transformations, 25

coordinate_system (base class), *see* NXcoordinate_system, 197
coordinate_system (field), 741
coordinate_system_set (base class), *see* NXcoordinate_system_set, 887
coordinates, 227
copyright, 1092
core_diameter (field), 926
core_index_of_refraction (field), 926
core_material (field), 926
core_sample_indices (field), 714
corrector_cs (base class), *see* NXcorrector_cs, 200
count (field), 189, 751, 752, 754, 755, 1007
count_time (field), 238, 336, 519, 1057
countrate_correction_applied (field), 241, 521
countrate_correction_lookup_table (field), 241, 521
coupled (field), 334
coupling_material (field), 334, 1020, 1029
crate (field), 238
creator (file attribute), 374
creator_version (file attribute), 374
critical_energy (field), 183
crystal (base class), *see* NXcrystal, 204
crystal_symmetry (field), 376
crystal_system (field), 416
crystallographic_calibration (field), 617

cs_computer (base class), *see* NXcs_computer, 209
cs_cpu (base class), *see* NXcs_cpu, 891
cs_filter_boolean_mask (base class), *see* NXcs_filter_boolean_mask, 891
cs_gpu (base class), *see* NXcs_gpu, 892
cs_io_obj (base class), *see* NXcs_io_obj, 893
cs_io_sys (base class), *see* NXcs_io_sys, 894
cs_memory (base class), *see* NXcs_memory, 210
cs_mm_sys (base class), *see* NXcs_mm_sys, 894
cs_prng (base class), *see* NXcs_prng, 895
cs_process (base class), *see* NXcs_process, 211
cs_profiling (base class), *see* NXcs_profiling, 212
cs_profiling_event (base class), *see* NXcs_profiling_event, 214
cs_storage (base class), *see* NXcs_storage, 215
csg (base class), *see* NXcsg, 896
cue_index (field), 283, 326
cue_timestamp_zero (field), 283, 326
cumulated (field), 794, 795
cumulated_normalized (field), 794, 795
current (field), 234, 303, 324, 396, 932
current_set_feature_to_cluster (field), 731
current_to_next_link (field), 730
current_to_next_link_type (field), 730
current_working_directory (field), 212, 633, 651, 658, 675, 683, 694, 701, 707, 710, 715, 724, 732, 760, 780, 795, 805, 818, 829, 965, 975
curvature (field), 421
curvature_horizontal (field), 207
curvature_radius_FACE (field), 938
curvature_vertical (field), 207
cut_angle (field), 205
cw (field), 836
cw_power (field), 836
cxix_ptycho (application definition), *see* NXcxix_ptycho, 897
cylinders (field), 216
cylindrical (field), 421
cylindrical_geometry (base class), *see* NXcylindrical_geometry, 216
cylindrical_orientation_angle (field), 207

D

d (field), 362
d_spacing (field), 206
data

- multi-dimensional**, 50

data (base class), *see* NXdata, 217
DATA (field), 231
data (field), 235, 269, 300, 336, 340, 429, 482, 506, 509, 511, 512, 514, 516, 519, 531, 533, 536, 539, 540, 543, 544, 546, 548–550, 552, 553, 555, 556, 558, 560, 561, 563, 564, 566, 569, 571, 573, 575, 581, 898, 899, 947, 1021, 1028, 1030

data analysis software, 1082
 data field, *see* field
 data group, *see* group
 data item, *see* field
 data object, *see* field
 data set, *see* field
 data type, 155
 data_correction (*field*), 921
 data_counts (*field*), 618, 619
 data_errors (*field*), 236
 data_identifier (*field*), 991
 data_offset (*field*), 517
 data_origin (*field*), 250, 523
 data_scaling_factor (*field*), 516
 data_size (*field*), 250, 523
 data_stride (*field*), 523
 data_type (*field*), 922, 991
 data_x_time_of_flight (*field*), 1028
 data_x_y (*field*), 1019, 1028
 data_y_time_of_flight (*field*), 1028
 dataset, *see* field
 dataset_name (*field*), 643, 663, 675, 706
 dataset_name_distances (*field*), 693, 694
 dataset_name_facet_indices (*field*), 651, 662, 670, 674
 dataset_name_facet_normals (*field*), 651, 670, 674
 dataset_name_facet_vertices (*field*), 671
 dataset_name_mass_to_charge (*field*), 639, 641, 642, 649, 661, 681, 687, 692, 699, 705
 dataset_name_reconstruction (*field*), 639, 641, 642, 649, 661, 681, 687, 692, 699, 705
 dataset_name_vertices (*field*), 651, 662, 670, 674
 date (*field*), 340, 358, 445, 569, 573, 909, 1019, 1027
 date and time, 45
 DAVE (*data analysis software*), 1083
 DAWN (*data analysis software*), 1083
 dead_time (*field*), 237, 519
 decorator_multiplicity (*field*), 756
 default, 47
 default (*file attribute*), 184, 277, 341, 371, 374, 389, 408
 default (*group attribute*), 433
 default attribute value, 47, 277, 374, 408
 default plot, *see* plotting
 default_slice (*file attribute*), 229
 definition (*field*), 277, 408, 425, 429, 433, 451, 454, 505, 507, 508, 511, 513, 516, 530, 532, 535, 539, 542, 544, 546, 548, 551, 555, 557, 560, 562, 565, 568, 570, 573, 574, 577, 579–582, 584, 601, 629, 640, 648, 654, 660, 680, 686, 690, 698, 704, 709, 711, 720, 728, 737, 776, 785, 792, 800, 811, 824, 897, 908, 919, 932, 933, 936, 942, 949, 961, 968, 983, 1009, 1017, 1026, 1043, 1051
 definition (*NXDL data type*), 143
 definition (*NXDL element*), 65, 136
 definition_local (*field*), 278, 408
 definitionType (*NXDL data type*), 143
 definitionTypeAttr (*NXDL data type*), 145
 deflection_angle (*field*), 297, 842
 deflector (*base class*), *see* NXdeflector, 234
 defocus (*field*), 345
 deformed_grain_identifier (*field*), 973
 defragment_cell_cache (*field*), 965
 defragment_x (*field*), 965
 delay (*field*), 253
 delay_difference (*field*), 1053
 delocalization (*base class*), *see* NXdelocalization, 902
 delta_time (*field*), 167
 density (*field*), 206, 290, 380, 386, 458, 886
 depends on (*field attribute*), 26
 depends_on (*field attribute*), 179, 180, 250, 251, 414, 524, 835
 depends_on (*field*), 169, 170, 175, 179, 182, 183, 188, 194, 196, 199, 207, 242, 251, 254, 264, 265, 281, 289, 291, 293, 295, 297, 299, 313, 332, 334, 336, 338, 355–357, 370, 391, 393, 398, 420, 422, 461, 463, 481, 482, 517, 519, 834–836, 988, 1004
 depends_on (*file attribute*), 958
 depolarization (*field*), 992
 deprecated, 171, 182, 184, 195, 207, 208, 232, 243, 254, 278, 289, 291, 295, 297, 300, 313, 326, 332, 334, 336–338, 382, 383, 391, 398, 420, 443, 525, 526, 1018, 1027, 1052
 depth (*field*), 297
 description (*field attribute*), 361–367, 917, 940, 1011
 description (*field*), 168, 171, 181, 185, 192, 196, 201, 214, 237, 255, 272, 281, 290, 296, 299, 303, 323, 325, 327, 331, 340, 353, 357, 370, 381, 386, 426, 445, 458, 519, 605, 607, 609, 611, 629, 634, 716, 725, 733, 761, 781, 788, 796, 806, 819, 830, 881, 886, 906, 907, 915, 917, 925, 940, 943, 976, 1005, 1013, 1018–1021, 1027, 1029, 1030, 1035, 1038, 1057
 description (*file attribute*), 360
 descriptor (*field*), 268, 482
 design (*field*), 318, 360, 468, 469, 480, 612
 design principles, 4
 det_module (*field*), 361
 details (*field*), 445
 detection_angle (*field*), 985
 detection_gas_path (*field*), 237
 detection_rate (*field*), 613
 detector (*base class*), *see* NXdetector, 234
 detector_channel (*base class*), *see* NXdetector_channel, 246
 detector_faces (*field*), 344

detector_group (*base class*), *see* NXdetector_group, 248
detector_identifier (*field*), 306, 400, 469, 472
detector_module (*base class*), *see* NXdetector_module, 250
detector_number (*field*), 216, 235, 555, 558
detector_readout_time (*field*), 241, 521
detector_type (*field*), 260, 944
detector_voltage (*field*), 260
diameter (*field*), 239, 355, 582, 924, 1049
dictionary of terms, 13
dielectric_function (*field*), 908, 910, 911
diffraction_order (*field*), 297
dim (*NXDL element*), 66
dimension, 19, 50
 dimension scales, 51, 53–55
 fastest varying, 42, 54, 251
 slowest varying, 42, 251
 storage order, 42
dimension scale, 23, 49
dimensionality (*field*), 416, 631, 643, 649, 650, 661, 662, 681, 692, 693, 699, 705, 741, 745, 746, 756–759, 802, 803, 816, 817, 846, 847, 849, 851, 855, 857, 859, 863, 866, 867, 869, 871, 874, 876, 878, 880, 929, 931, 958, 971
dimensions (*NXDL element*), 66, 136
dimensionsType (*NXDL data type*), 145
direction, *see* vector (*field attribute*)
direction (*field attribute*), 379
direction (*field*), 393
direction_model (*field*), 674
directionality (*field*), 928
directctof (*application definition*), *see* NXdirectctof, 450
disk_chopper (*base class*), *see* NXdisk_chopper, 252
dislocation_density (*field*), 974
disorientation_angle (*field*), 377
disorientation_axis (*field*), 377
disorientation_quaternion (*field*), 377
dispersion (*base class*), *see* NXdispersion, 903
dispersion (*field*), 258, 476, 925
dispersion_function (*base class*), *see* NXdispersion_function, 904
dispersion_repeated_parameter (*base class*), *see* NXdispersion_repeated_parameter, 906
dispersion_single_parameter (*base class*), *see* NXdispersion_single_parameter, 906
dispersion_table (*base class*), *see* NXdispersion_table, 907
dispersion_type (*field*), 909, 925
dispersive_material (*application definition*), *see* NXdispersive_material, 908
DispersiveMaterial, 591
distance (*field*), 174, 176, 237, 253, 288, 333, 336, 382, 394, 442, 508, 511, 519, 533, 536, 539, 544, 555, 556, 558, 560, 561, 563, 566, 575, 722, 794, 795, 899, 943, 1019–1021, 1028–1030, 1057
distance_derived (*field*), 519
distance_to_detector (*field*), 182
distances (*field*), 415
distancing_model (*field*), 673
distortion (*base class*), *see* NXdistortion, 255
distribution (*field attribute*), 231
divergence (*field*), 836
divergence_x (*field*), 194
divergence_x_minus (*field*), 183
divergence_x_plus (*field*), 183
divergence_y (*field*), 194
divergence_y_minus (*field*), 183
divergence_y_plus (*field*), 183
doc (*NXDL element*), 66, 136
docType (*NXDL data type*), 147
documentation editor, 1091
doi (*field*), 192, 909
domain_identifier (*field*), 965
domain_size (*field*), 961
dose_management (*field*), 346
dose_rate (*field*), 346
download location, *see* NAPI installation
dQ1 (*field*), 440
dQw (*field*), 440
drain_current (*field*), 945
dspacing (*field*), 268
duration (*field*), 278, 326, 409, 425, 557, 560, 1017, 1018, 1026, 1027
duty_cycle (*field*), 297
dwell_time (*field*), 388, 479
dynamic_focus_correction (*field*), 346
dynamic_phi_list (*field*), 1058
dynamic_q_list (*field*), 1058
dynamic_refocusing (*field*), 346
dynamic_roi_map (*field*), 1058

E

ebeam_column (*base class*), *see* NXebeam_column, 256
edge_contact (*field*), 759
edge_distance (*field*), 693
edge_handling_method (*field*), 664
edge_identifier (*field*), 852
edge_identifier_offset (*field*), 756, 852
edge_length (*field*), 747, 756, 758, 867, 870, 876, 878
edge_threshold (*field*), 665
edges (*field*), 852
edges_are_unique (*field*), 853
ef (*field*), 552
efficiency (*field*), 243, 336, 356, 838
ei (*field*), 552

elapsed_time (*field*), 214, 634, 716, 725, 733, 761, 781, 788, 796, 806, 819, 830, 976
electric_field (*field*), 379, 427
electrical_conductivity (*field*), 936
electron_detector (*base class*), *see* NXelectron_detector, 260
electronanalyser (*base class*), *see* NXelectronanalyser, 915
electrostatic_kicker (*base class*), *see* NXelectrostatic_kicker, 917
element_whitelist (*field*), 903
ellipsometer_type (*field*), 920
Ellipsometry, 591
ellipsometry (*application definition*), *see* NXellipsometry, 918
em (*application definition*), *see* NXem, 451
em_ebsd (*base class*), *see* NXem_ebsd, 261
em_eds (*base class*), *see* NXem_eds, 270
em_eels (*base class*), *see* NXem_eels, 273
em_img (*base class*), *see* NXem_img, 274
em_measurement (*base class*), *see* NXem_measurement, 275
em_simulation (*base class*), *see* NXem_simulation, 276
email (*field*), 418, 456, 603, 631, 712, 721, 729, 738, 777, 786, 793, 801, 812, 825, 942, 951, 970, 984, 1010, 1044
EmAppDef, 588
EmBC, 588
embedding_medium (*field*), 936
emission_current (*field*), 257, 397, 476
emittance_x (*field*), 396
emittance_y (*field*), 396
emitter_material (*field*), 257
emitter_type (*field*), 257, 303, 465, 467
en (*field*), 547, 552
end_time (*field*), 168, 213, 214, 278, 285, 335, 409, 425, 455, 469, 516, 530, 532, 535, 542, 548, 562, 565, 602, 630, 633, 634, 712, 715, 716, 721, 724, 725, 729, 732, 733, 737, 760, 761, 777, 780, 781, 786, 788, 792, 795, 796, 800, 805, 806, 811, 818, 819, 825, 829, 830, 897, 934, 936, 950, 969, 975, 976, 1009, 1017, 1026, 1052
end_time_estimated (*field*), 516
endnote (*field*), 193
energies (*field*), 429
energy (*field*), 272, 288, 313, 338, 396, 429, 451, 506, 507, 544, 548, 550, 571, 573, 898, 908
energy_dispersion (*field*), 338
energy_error (*field*), 338
energy_errors (*field*), 338
energy_identifier (*field*), 904, 910, 911
energy_interval (*field*), 924
energy_max (*field*), 904
energy_min (*field*), 904
energy_range (*field*), 272, 483
energy_resolution (*field*), 915, 942, 943
energy_scan_mode (*field*), 924, 944
energy_transfer (*field*), 176
energy_unit (*field*), 904, 910–912
energydispersion (*base class*), *see* NXenergydispersion, 923
entering (*field*), 361
entrance_slit_setting (*field*), 429
entrance_slit_shape (*field*), 429
entrance_slit_size (*field*), 429
entry (*base class*), *see* NXentry, 276
entry_identifier (*field*), 277, 408, 425, 1017, 1026, 1051
entry_identifier_uuid (*field*), 277, 1051
enumeration, 46
enumeration (*NXDL element*), 138
enumerationType (*NXDL data type*), 147
environment (*base class*), *see* NXenvironment, 281
EPICS
 instrument examples, 117
epoch_start (*field attribute*), 264
eps (*field*), 644, 713
equipment_component (*field attribute*), 414
error (*field*), 544
errors (*field*), 232
estimated (*field attribute*), 168
estimated_field_at_the_apex (*field*), 613
euler (*field*), 974
euler_angle_convention (*field*), 458, 889, 951, 971
eulerian_cradle, 26, 424, 577
evaporation_id_included (*field*), 615
even_layer_density (*field*), 332
even_layer_material (*field*), 332
event_data (*base class*), *see* NXevent_data, 282
event_data_em (*base class*), *see* NXevent_data_em, 284
event_id (*field*), 283
event_index (*field*), 283, 1019
event_pixel_id (*field*), 1019
event_time_of_flight (*field*), 1020
event_time_offset (*field*), 283
event_time_zero (*field*), 283
examples
 Nexus file, 4
 Nexus file; minimal, 6
excitation_wavelength (*field*), 836
experiment_alias (*field*), 454
experiment_description (*field*), 277, 408, 425, 454, 601, 920, 983, 1009, 1044
experiment_identifier (*field*), 277, 408, 425, 601, 983, 1009, 1017, 1026, 1044
experiment_or_simulation (*field*), 950, 969
experiment_type (*field*), 920, 983
experiments (*field*), 361

- `exposure_time (field)`, 201
`extends (NXDL attribute)`, 65
`extent (field)`, 177, 632, 741, 855, 898, 972, 1056
`external_DAC (field)`, 382
`external_field_brief (field)`, 390
`external_material (field)`, 299, 331
`extinction_ratio (field)`, 998
`extraction_voltage (field)`, 257, 476
`extractor_current (field)`, 883
`extractor_voltage (field)`, 883
- F**
- `fabrication (base class)`, *see* NXfabrication, 287
`fabrication (field)`, 294
`face_area (field)`, 860, 869, 876
`face_half_edge (field)`, 857
`face_identifier (field)`, 757, 758, 852
`face_identifier_offset (field)`, 743, 746, 757, 758, 804, 817, 852, 857
`face_normals (field)`, 751, 753–755
`faces (field)`, 344, 743, 747, 751, 752, 754, 755, 757, 758, 804, 852
`faces_are_unique (field)`, 853
`facility_user_id (field)`, 418, 426, 1022, 1031
FAQ, 73
`fast_axes (field)`, 916, 943
`fast_pixel_direction (field)`, 251, 524
`fax_number (field)`, 418
FDL, 1092
`feature_identifier (field)`, 657, 658, 715, 750–754, 1015
`feature_member_count (field)`, 715, 1015
`feature_type (field)`, 656, 657, 749
`feature_type_dict_keyword (field)`, 749
`feature_type_dict_value (field)`, 749
`features (field)`, 277
`feed_forward_model (field)`, 354
fermi_chopper (*base class*), *see* NXfermi_chopper, 288
fiber (*base class*), *see* NXfiber, 925
`field`, 4, 18
 HDF, 55
`field (NXDL element)`, 66, 138
`field attribute`, 4, 14, 18
`field_of_view (field)`, 345, 608
`FIELDNAME_errors (field)`, 231, 341
`FIELDNAME_mask (field)`, 341
`FIELDNAME_offset (field)`, 232
`FIELDNAME_scaling_factor (field)`, 232
`FIELDNAME_set (field)`, 341
`FIELDNAME_weights (field)`, 341
`fieldType (NXDL data type)`, 148
`figure_of_merit (field)`, 1039
`filament_current (field)`, 257, 397, 476
`file`
- read and write, 13
 validate, 1082
`file attribute`, 19
`file format`, 55
 HDF, 55
`file_name (field)`, 340, 455, 469, 473, 481, 482, 670
`file_name (file attribute)`, 374
`file_time (file attribute)`, 373
`file_update_time (file attribute)`, 374
`filename (field)`, 630, 638, 639, 641, 642, 649, 651, 657, 658, 661–663, 672, 674, 680, 681, 683, 687, 692–694, 699, 705, 706, 710, 962
`filenames (field)`, 509, 546
`filter (base class)`, *see* NXfilter, 289
`final_beam_divergence (field)`, 178
`final_energy (field)`, 176
`final_polarization (field)`, 177
`final_polarization_stokes (field)`, 178
`final_wavelength (field)`, 177
`final_wavelength_spread (field)`, 178
`find the default plottable data`, 47, 229, 277, 374, 408
`first_good (field attribute)`, 231
`first_point_for_fit (field attribute)`, 1055
`fit_formula_description (field)`, 185
`fixed_energy (field)`, 544
`fixed_revolutions (field)`, 922
`fixed_slit (field)`, 839
`flags (field)`, 361
`flatfield (field)`, 239, 247, 520
`flatfield_applied (field)`, 239, 247, 520
`flatfield_error (field)`, 520
`flatfield_errors (field)`, 239, 247, 520
`flexible name`, 133
`flight_path_length (field)`, 608
`flip_current (field)`, 293
`flip_turns (field)`, 293
`flipper (base class)`, *see* NXflipper, 293
`floating-point numbers`, 45
`fluence (field)`, 178
`fluo (application definition)`, *see* NXfluo, 505
`flux (field)`, 178, 303, 396, 479, 525
`flux (group attribute)`, 524
`flux_integrated (field)`, 526
`flyback_time (field)`, 388
`focal_length (field)`, 346, 938
`focal_size (field)`, 188
`focus_parameters (field)`, 294
`focus_type (field)`, 421
`folder`, *see* group
`force (field)`, 934
`force_control (field)`, 934
`format unification`, 12
`formula (field)`, 904, 910, 911

four-circle diffractometer, 26, 35, 424, 577
 frame_average (*field*), 1052
 frame_start_number (*field*), 239, 575
 frame_sum (*field*), 1052
 frame_time (*field*), 241, 521, 1057
 frequency (*field attribute*), 235
 frequency (*field*), 194, 396, 1019, 1028
 frequency_resolution (*field*), 838
 fresnel_zone_plate (*base class*), *see* NXfresnel_zone_plate, 294
 frog_delays (*field*), 179
 frog_frequencies (*field*), 179
 frog_trace (*field*), 178

G

g2 (*field*), 1052
 g2_derr (*field*), 1053
 g2_err_from_two_time_corr_func (*field*), 1055
 g2_err_from_two_time_corr_func_partials (*field*), 1056
 g2_from_two_time_corr_func (*field*), 1055
 g2_from_two_time_corr_func_partials (*field*), 1056
 G2_unnormalized (*field*), 1053
 gain (*field*), 190, 1046
 gain_setting (*field*), 241, 522
 gamma (*field*), 416, 482
 gap (*field*), 313
 gas (*field*), 422
 gas_pressure (*field*), 237, 397, 422, 946
 GDA (*data acquisition software*), 1083
 geometry, 25, 30
 geometry (*base class*), *see* NXgeometry, 295
 geometry (*field*), 896
 git, 1075
 gradient_guide_magnitude (*field*), 748
 gradient_guide_projection (*field*), 748
 grain_diameter (*field*), 604, 962
 grain_diameter_error (*field*), 604
 grain_euler (*field*), 962
 grain_identifier (*field*), 973
 grain_size (*field*), 962
 graph_edge_set (*base class*), *see* NXgraph_edge_set, 928
 graph_node_set (*base class*), *see* NXgraph_node_set, 929
 graph_root (*base class*), *see* NXgraph_root, 930
 grating (*base class*), *see* NXgrating, 296
 grating_wavelength_max (*field*), 839
 grating_wavelength_min (*field*), 838
 grid (*field*), 902
 gridresolutions (*field*), 664
 grooves (*field*), 838
 group, 4, 17

HDF, 55
 group (*NXDL element*), 65, 139
 group_attribute, 4, 18
 group_index (*field*), 249, 518
 group_name_iotypes (*field*), 638, 642, 649, 661, 681, 683, 687, 692, 699, 705
 group_names (*field*), 249, 518
 group_parent (*field*), 249, 518
 group_type (*field*), 249
 groupname_geometry_prefix (*field*), 657, 658
 groupType (*NXDL data type*), 151
 guide (*base class*), *see* NXguide, 298
 guide_current (*field*), 293
 guide_turns (*field*), 293
 Gumtree (*data analysis software*), 1083

H

h (*field*), 361
 h5py
 code examples, 82
 h5py_version (*file attribute*), 374
 h5wasm
 tools, 1084
 h5web
 tools, 1084
 hagb_enthalpy (*field*), 963
 hagb_pre_factor (*field*), 963
 half_angle_interval (*field*), 275
 half_axes_radii (*field*), 643, 649, 661, 681, 687, 692, 699, 705, 850
 half_axes_radius (*field*), 850
 half_edge_identifier_offset (*field*), 857
 half_edge_incident_face (*field*), 857
 half_edge_next (*field*), 858
 half_edge_prev (*field*), 858
 half_edge_twin (*field*), 857
 half_edge_vertex_origin (*field*), 857
 halo_region (*field*), 973
 harmonic (*field*), 313
 has_cell_edge_detection (*field*), 706
 has_cell_geometry (*field*), 706
 has_cell_neighbors (*field*), 706
 has_cell_volume (*field*), 706
 has_closure (*field*), 701
 has_current_to_next_links (*field*), 656
 has_exterior_facets (*field*), 701
 has_interior_tetrahedra (*field*), 701
 has_next_to_current_links (*field*), 656
 has_object (*field*), 666
 has_object_auto_proxigram (*field*), 668
 has_object_auto_proxigram_edge_contact (*field*), 668
 has_object_edge_contact (*field*), 667
 has_object_geometry (*field*), 666

has_object_ions (*field*), 667
 has_object_obb (*field*), 667
 has_object_properties (*field*), 667
 has_proxy (*field*), 667
 has_proxy_edge_contact (*field*), 668
 has_proxy_geometry (*field*), 668
 has_proxy_ions (*field*), 668
 has_proxy_obb (*field*), 668
 has_proxy_properties (*field*), 668
 has_scalar_fields (*field*), 664
 has_triangle_soup (*field*), 666
 HDF
 file format, 55
 tools, 1084
 HDF4, 1088
 HDF5, 1088
 HDF5_Version (*file attribute*), 374
 HDF_version (*file attribute*), 374
 HDFExplorer
 tools, 1084
 HDFview
 tools, 1084
 heat_treatment_quenching_rate (*field*), 604
 heat_treatment_quenching_rate_error (*field*), 605
 heat_treatment_temperature (*field*), 604
 heat_treatment_temperature_error (*field*), 604
 heater_current (*field*), 480
 heater_power (*field*), 480
 heater_voltage (*field*), 480
 height (*field*), 288, 420, 643, 650, 662, 681, 687, 692, 699, 705, 848, 860
 hierarchy, 4, 17, 18, 31, 33, 62, 63, 69, 1081
 high_throughput_method (*field*), 644, 645
 high_trip_value (*field*), 390
 histogram_min_incr_max (*field*), 694
 history (*base class*), *see NXhistory*, 301
 hit_multiplicity (*field*), 615
 hit_positions (*field*), 614
 holder (*field*), 1021, 1030

|

I (*field*), 437
 I_axes (*group attribute*), 434
 ibeam_column (*base class*), *see NXbeam_column*, 302
 IcmeMsModels, 596
 id (*field*), 322, 361
 id_type (*field*), 322
 identifier (*field*), 608–612, 633, 634, 643, 650, 662, 681, 693, 700, 706, 716, 724, 732, 733, 760, 761, 781, 788, 795, 796, 805, 806, 817, 818, 829, 830, 848, 849, 855, 860, 864, 866, 867, 870, 872, 874, 876, 878, 892, 928, 930, 934, 937, 959, 975, 976, 997, 1021, 1030, 1044
 identifier_calibration_method (*field*), 185

identifier_calibration_reference (*field*), 185
 identifier_canonical_smiles (*field*), 1042
 identifier_cas (*field*), 1041
 identifier_chemical (*field*), 172
 identifier_event (*field*), 285
 identifier_experiment (*field*), 454
 identifier_inchi_key (*field*), 1042
 identifier_inchi_str (*field*), 1042
 identifier_iupac_name (*field*), 1042
 identifier_offset (*field*), 632, 643, 649, 650, 661, 662, 681, 692, 693, 699, 705, 714, 742, 746, 756, 758, 803, 804, 817, 848, 849, 855, 860, 864, 866, 867, 870, 872, 874, 876, 878, 882, 928, 929, 953, 959, 972, 982, 996, 1015
 identifier_parent (*field*), 457
 identifier_phase (*field*), 266
 identifier_pub_chem (*field*), 1042
 identifier_sample (*field*), 285, 457, 469
 identifier_smiles (*field*), 1042
 identifierNAME (*field*), 301, 340, 341, 455
 Idev (*field*), 439
 IDF_Version (*file attribute*), 277, 408
 IDL (*data analysis software*), 1083
 IGOR Pro (*data analysis software*), 1083
 imag (*field*), 306–309, 469–472
 image (*base class*), *see NXimage*, 304
 image_key (*field*), 240, 563
 images, 45
 imaging_mode (*field*), 275, 481
 implementation (*field*), 746, 862
 incident_angle (*field*), 299, 331, 842
 incident_beam_divergence (*field*), 177, 898
 incident_beam_energy (*field*), 898
 incident_beam_size (*field*), 526
 incident_energy (*field*), 176, 943, 1056
 incident_energy_spread (*field*), 176, 898, 943, 1056
 incident_energy_weights (*field*), 176
 incident_polarisation_stokes (*field*), 526
 incident_polarization (*field*), 177, 943
 incident_polarization_stokes (*field*), 177, 526
 incident_polarization_type (*field*), 1056
 incident_wavelength (*field*), 176, 444, 524
 incident_wavelength_spread (*field*), 177, 444, 525
 incident_wavelength_weight (*field*), 525
 incident_wavelength_weights (*field*), 177, 525
 incubation_time_model (*field*), 962
 independent_controllers (*field*), 1010
 index (*group attribute*), 425
 index (*NXDL attribute*), 66
 index_of_refraction (*field*), 939, 1000
 index_of_refraction_coating (*field*), 844, 939, 1000, 1050
 index_of_refraction_substrate (*field*), 844, 1050
 indexing_rate (*field*), 267, 482

indices (*field*), 173
indices_group (*field*), 308–310, 403, 404, 471, 472
indices_image (*field*), 308–310, 471, 472
indices_spectrum (*field*), 403, 404, 474, 475
indirectttof (*application definition*), see NXindirectttof, 507
infection_direction (*field*), 974
ingestion, 1081
initial_cell_cache (*field*), 964
initial_radius (*field*), 613
initialization (*field*), 671
input (*field*), 238, 663
input_impedance (*field*), 190
input_path (*field attribute*), 185, 186
inputs (*field*), 196
insertion_device (*base class*), *see* NXinser-
tion_device, 312
inspection, 1081
installation, *see* NAPI installation
instrument (*base class*), *see* NXinstrument, 314
instrument definitions, 8
instrument_em (*base class*), *see* NXinstrument_em, 316
instrument_name (*field*), 608
int_prf (*field*), 366
int_prf_errors (*field*), 366
int_prf_var (*field*), 366
int_sum (*field*), 366
int_sum_errors (*field*), 366
int_sum_var (*field*), 366
integers, 45
integral (*field*), 336, 514, 533, 537, 566, 576
integral_counts (*field*), 558
integration_radius (*field*), 954
intensity (*field*), 271, 306–309, 351, 401–404, 470–
475, 483, 744, 745
interaction_volume_em (*base class*), *see* NXinterac-
tion_volume_em, 320
interface_model (*field*), 668
interior_angle (*field*), 748, 756, 758, 867, 878
interior_atmosphere (*field*), 297, 299, 331
interpretation (*field attribute*), 898
intersection_detection_method (*field*), 655
intersection_volume (*field*), 731
introduction, 3
IntroductionApm, 594
IntroductionCgms, 596
IntroductionEm, 588
IntroductionMpes, 593
IntroductionTransport, 595
ion (*base class*), *see* NXion, 321
ion_energy_profile (*field*), 303
ion_identifier (*field*), 752–755
ion_multiplicity (*field*), 755
ion_query_isotope_vector (*field*), 642
ion_query_isotope_vector_source (*field*), 691
ion_query_isotope_vector_target (*field*), 692
ion_query_type_source (*field*), 691
ion_query_type_target (*field*), 691
ion_type (*field*), 322
ion_type_filter (*field*), 641
iontypes (*field*), 778
iontypes_randomized (*field*), 794
iqproc (*application definition*), *see* NXiqproc, 508
is_a (*field*), 929, 930
is_antipodal (*field*), 377
is_axis_aligned (*field*), 742, 860, 864
is_box (*field*), 860
is_centrosymmetric (*field*), 417
is_chiral (*field*), 417
is_closed (*field*), 850, 872, 875
is_core (*field*), 715
is_cylindrical (*field*), 207
is_deformed (*field*), 974
is_noise (*field*), 714
is_polycrystalline (*field*), 607
is_recrystallized (*field*), 974
is_simulation (*field*), 456
is_specific (*field*), 1016
is_watertight (*field*), 803
ISAW (*data analysis software*), 1083
ISO8601 (*data type*), 155
isocontour (*base class*), *see* NXisocontour, 931
isotope (*field*), 760
isotope_matrix (*field*), 828
isotope_vector (*field*), 620, 751, 752, 778, 780, 828
isotope_vector_matrix (*field*), 779
isotope_whitelist (*field*), 663, 682, 903
isovalue (*field*), 746, 931
issue reporting, 1075
iteration (*field*), 953, 973, 981
iupac_line_candidates (*field*), 272, 483
iupac_line_name (*field*), 272
iv_temp (*application definition*), *see* NXiv_temp, 932

J

job_pyiron_identifier (*field*), 629
Jones_quality_factor (*field*), 993
jupyterlab-h5web
tools, 1084

K

k (*field*), 313, 361
K_d (*field*), 354
K_ff (*field*), 354
K_i (*field*), 353
K_p (*field*), 353
kappa (*field*), 579
kernel_mu (*field*), 742

kernel_sigma (field), 742
kernel_size (field), 664, 742
kernel_variance (field), 664
ki_over_kf_scaling (field), 544
Kłosowski, Przemysław, 11, 1088
Könnecke, Mark, 11, 1089

L

l (field), 361
lab_electro_chemo_mechanical_preparation (application definition), *see* NXlab_electro_chemo_mechanical_preparation, **933**
lab_sample_mounting (application definition), *see* NXlab_sample_mounting, **936**
label (field), 351, 620, 929, 930
lagb_enthalpy (field), 963
lagb_pre_factor (field), 963
lagb_to_hagb_cut (field), 963
lagb_to_hagb_exponent (field), 963
lagb_to_hagb_transition (field), 963
lambda (field), 447
LAMP (data analysis software), 1083
last_fill (field), 397
last_good (field attribute), 231
last_process (field), 185
lateral_surface_area (field), 848
lattice_type (field), 1016
laue_group (field), 416
lauetof (application definition), *see* NXlauetof, **510**
layer_structure (field), 990
layer_thickness (field), 297, 332
layout (field), 238
length (field), 313, 419, 442, 860, 864, 872, 925
lens_diameter (field), 938
lens_em (base class), *see* NXlens_em, **323**
lens_geometry (field), 421
lens_length (field), 421
lens_material (field), 422
lens_mode (field), 429
lens_opt (base class), *see* NXlens_opt, **937**
lens_thickness (field), 421
lexicography, 13
LGPL, 1092
license, 1092
lifespan (field), 835
lifetime (field), 257
linear_range_min_incr_max (field), 688, 700, 1040
link, 4, **20**, 51, **75**, 161
 external file, 21, 22
link (NXDL element), **139**
link target, 139
link target (internal attribute), **19**
link, target, attribute, 20

linkType (NXDL data type), **152**
local_name (field attribute), 238
local_name (field), 237
location (field), 317, 464, 608
log (base class), *see* NXlog, **324**
long_name (field attribute), 231, 235–237, 269, 271, 272, 306–310, 401–405, 469–476, 482, 483, 618, 619
long_name (group attribute), **744**, **745**
low_trip_value (field), 390
low-level file format, *see* file format
lower_cap_radius (field), 848
lp (field), 366
LRMECS instrument examples, **114**
lubricant (field), 934

M

m_value (field), 291, 299, 331
Mac OS X, *see* NAPI installation
magnetic_field (field), 183, 379, 427
magnetic_kicker (base class), *see* NXmagnetic_kicker, **940**
magnetic_wavelength (field), 313
magnification (field), 201, 345, 883
magnitude (field), 167, 469–472, 476–478
magnitude_errors (field), 167
magnitude_errors_model (field), 167
mailing lists, **1074**
manipulator (base class), *see* NXmanipulator, **327**
Mantid (data analysis software), **1083**
manual source, 1091
manufacturer (field), 188
manufacturer_name (field), 609
mapping_MAPPING (field), 186
mark (field), 903
mask (field), 643, 650, 662, 681, 693, 700, 706, 713, 722, 723, 739, 779, 787, 794, 802, 803, 813–816, 892
mask (group attribute), **434**
Mask_indices (group attribute), **434**
mask_material (field), 294
mask_thickness (field), 294
mass (field), 380, 386, 779
mass_to_charge (field), 616, 828
mass_to_charge_interval (field), 682
mass_to_charge_range (field), 322, 620, 778, 780, 828
mass_to_charge_state_ratio (field), 779
mass_vector (field), 828
match (field), 662, 674, 688, 700, 740, 741, 941
match_filter (base class), *see* NXmatch_filter, **941**
matching_phase (field), 267
matching_phase_descriptor (field), 267
material (field), 171, 836, 989
material_phase (field), 909

material_phase_comment (field), 909
MATLAB, 1083
 code examples, 101
max_delta_x (field), 964
max_eps (field), 645
max_frequency (field), 838
max_gap (field), 839
max_iteration (field), 964
max_physical_capacity (field), 211, 215, 634, 716, 724, 733, 761, 781, 788, 796, 805, 818, 830, 893, 976
max_resident_memory_snapshot (field), 215, 634, 716, 725, 733, 761, 781, 789, 796, 806, 819, 830, 976
max_revolutions (field), 922
max_time (field), 964
max_virtual_memory_snapshot (field), 215, 634, 716, 725, 733, 761, 781, 789, 796, 806, 819, 830, 976
max_x (field), 964
maximum_charge (field), 682
maximum_incident_angle (field), 188
maximum_number_of_atoms_per_molecular_ion (field), 619, 778, 780
maximum_number_of_isotopes (field), 682
maximum_value (field), 326, 1018, 1027
McStas, 30
measure_time (field), 835
measured_data (field), 991, 1045
measured_data_errors (field), 992
measurement (field), 327–329, 389
measurement_sensors (field), 1011
medium (field), 986
medium_refractive_indices (field), 986
melting_temperature (field), 963
metadata, 23, 62, 63, 69, 73, 135
method (field), 266, 650, 662, 672, 674, 688, 700, 706, 740, 741, 941
Microsoft Windows, *see* NAPI installation
miller (field), 268
miller_direction (field), 1016
miller_plane (field), 1016
mime_type (group attribute), 409
min_abundance_product (field), 829
min_cluster_size (field), 645
min_frequency (field), 838
min_half_life (field), 829
min_pts (field), 644, 645, 714
min_samples (field), 645
minimum_value (field), 326, 1018, 1027
mirror (base class), *see* NXmirror, 331
misorientation_angle (field), 377
misorientation_axis (field), 377
misorientation_quaternion (field), 377
mobility_weight (field), 973
mode (field), 324, 335, 397, 400, 479, 506, 511, 514, 531, 533, 537, 540, 553, 556, 558, 561, 571, 576, 802, 846, 883, 944, 1021, 1030
model (field), 201, 287, 389, 464–469, 608–612, 934, 937, 963, 964, 984, 1044
model_labels (field), 714
model_name (field), 903, 904, 907, 910, 911
moderator (base class), *see* NXmoderator, 333
module_offset (field), 250, 523
molecular_formula_hill (field), 1041
molecular_mass (field), 1041
momentum_resolution (field), 915
monitor, 46
monitor (base class), *see* NXmonitor, 335
monochromator (base class), *see* NXmonochromator, 337
monopd (application definition), *see* NXmonopd, 513
mosaic_horizontal (field), 207
mosaic_vertical (field), 207
motivation, 3, *see* dictionary of terms, *see* exchange format, *see* format unification, *see* plotting, 11, 23
mounting_method (field), 936
mpes (application definition), *see* NXmpes, 941
MpesAppDef, 593
MpesBC, 593
MpesCommonBC, 593
MpesExtendedBC, 593
ms (application definition), *see* NXms, 949
ms_feature_set (base class), *see* NXms_feature_set, 958
ms_score_config (application definition), *see* NXms_score_config, 960
ms_score_results (application definition), *see* NXms_score_results, 968
ms_snapshot (base class), *see* NXms_snapshot, 981
ms_snapshot_set (base class), *see* NXms_snapshot_set, 981
multi-dimensional data, 50
multi-modal data, 161
multiple_outputs (field), 842
mx (application definition), *see* NXmx, 515

N

name (field attribute), 433
name (field), 167, 169, 172, 196, 209, 234, 281, 303, 314, 317, 323, 327, 352, 357, 378, 385, 389, 394, 418, 426, 429, 430, 442, 444, 445, 456, 457, 464–468, 482, 506, 509, 511, 513, 514, 517, 527, 530, 531, 533, 535, 537, 539, 540, 544, 546, 548, 552, 555, 557, 558, 560, 561, 563, 566, 568, 570, 571, 573–575, 603, 605, 609–611, 620, 630, 632–634, 712, 715, 716, 721, 724, 729, 732, 733, 738, 760, 761, 777, 780, 781, 786, 788, 793, 795, 796, 801, 805, 812,

818, 825, 829, 830, 881, 886, 891, 893, 898,
 900, 906, 907, 910–912, 915, 942, 943, 946,
 950, 951, 954, 970, 975, 976, 983, 1002, 1010,
 1011, 1019, 1021, 1022, 1028, 1031, 1041,
 1044, 1047
name (*group attribute*), 446
name (*NXDL attribute*), 65, 66
NAME_spectrum (*field*), 991
naming convention, 38
NAPI, 4, 13, 13, 75, 1061, 1081, 1088
 bypassing, 55
 c, 1064
 c++, 1064
 core, 1063
 f77, 1064
 f90, 1064
 IDL, 1070
 installation, 1076
 installation; download location, 1077
 installation; Mac OS X, 1078
 installation; RPM, 1077
 installation; source distribution, 1078
 installation; Windows, 1078
 java, 1065
natural_abundance_product (*field*), 779
natural_abundance_product_vector (*field*), 828
nature (*field*), 558, 561, 1022, 1031
ndattribute, 121
Nelson, Mitchell, 11
NeXpy, 7
NeXpy (*data analysis software*), 1083
next_set_feature_to_cluster (*field*), 731
next_to_current_link (*field*), 730
next_to_current_link_type (*field*), 730
NeXus Application Programming Interface, *see*
 NAPI
NeXus Constructor, 1082
NeXus Definition Language, *see* NXDL
NeXus International Advisory Committee, *see*
 NIAC
NeXus link, 20, 20, 22
NeXus webpage, 1073
NeXus_release (*file attribute*), 374
NeXus_repository (*file attribute*), 374
NeXus_version (*file attribute*), 374
NIAC, 9, 1073, 1073, 1088
node_pair (*field*), 928
noise (*field*), 882, 1015
noise_level (*field*), 191
nominal (*field*), 336
nonNegativeUnbounded (*NXDL data type*), 155
normalization (*field*), 189, 605, 664, 739
normals (*field*), 748, 757–759, 880
note (*base class*), *see* NXnote, 339
notes (*field*), 1017, 1026
nth (*field*), 694
nucleus_euler (*field*), 962
nuclid_list (*field*), 620, 751, 752, 778, 780, 828
nuclide_hash (*field*), 322
nuclide_list (*field*), 322
num (*field*), 419
number (*field*), 288
number_of_atoms (*field*), 759
number_of_boundaries (*field*), 744, 855, 952, 972
number_of_bunches (*field*), 396
number_of_categorical_labels (*field*), 882, 1015
number_of_cluster (*field*), 882
number_of_core (*field*), 715
number_of_cycles (*field*), 242
number_of_disjoint_nuclids (*field*), 780
number_of_domains (*field*), 965
number_of_edges (*field*), 851, 869, 928
number_of_faces (*field*), 746, 756, 758, 803, 804, 817,
 851, 857, 869
number_of_feature_types (*field*), 656, 657
number_of_features (*field*), 715, 1015
number_of_files (*field*), 651
number_of_gpus (*field*), 213, 214, 633, 634, 715, 716,
 724, 725, 732, 733, 760, 761, 780, 781, 788,
 789, 795, 796, 805, 806, 818, 819, 829, 830,
 975, 976
number_of_half_edges (*field*), 857
number_of_ion_types (*field*), 618
number_of_ions (*field*), 713, 722, 739, 760, 779, 787,
 794, 801, 803, 812–816
number_of_iterations (*field*), 672
number_of_lenses (*field*), 421
number_of_noise (*field*), 715
number_of_nuclids (*field*), 780
number_of_numeric_labels (*field*), 882, 1014
number_of_objects (*field*), 643, 650, 662, 681, 693,
 700, 706, 892
number_of_parameters (*field*), 987
number_of_parent_identifier (*field*), 959
number_of_planes (*field*), 268
number_of_points (*field*), 723
number_of_processes (*field*), 213, 214, 633, 634, 641,
 649, 655, 661, 680, 687, 690, 699, 705, 715,
 716, 724, 725, 732, 733, 760, 761, 780, 781,
 788, 789, 795, 796, 805, 806, 818, 819, 829,
 830, 975, 976
number_of_scan_points (*field*), 267, 268, 482
number_of_threads (*field*), 213, 214, 633, 634, 715,
 716, 724, 725, 732, 733, 760, 761, 780, 781,
 788, 789, 795, 796, 805, 806, 818, 819, 829,
 830, 975, 976
number_of_total_vertices (*field*), 867, 871
number_of_triangles (*field*), 723

number_of_unassigned_members (*field*), 1015
number_of_unique_vertices (*field*), 878
number_of_vertices (*field*), 746, 756, 758, 803, 804,
 817, 851, 857, 871
number_sections (*field*), 299
numbers, *see* floating-point numbers, *see also* integers
numeric_label (*field*), 882
numerical_aperture (*field*), 926
numerical_label (*field*), 714, 1015
NX
 used as NX class prefix, 22, 38
NX_ANGLE (*units type*), 157
NX_ANY (*units type*), 157
NX_AREA (*units type*), 157
NX_BINARY (*data type*), 156
NX_BOOLEAN (*data type*), 156
NX_CCOMPLEX (*data type*), 156
NX_CHAR (*data type*), 156
NX_CHAR_OR_NUMBER (*data type*), 156
NX_CHARGE (*units type*), 157
NX_COMPLEX (*data type*), 156
NX_COUNT (*units type*), 157
NX_CROSS_SECTION (*units type*), 157
NX_CURRENT (*units type*), 157
NX_DATE_TIME (*data type*), 156
NX_DIMENSIONLESS (*units type*), 157
NX_EMITTANCE (*units type*), 157
NX_ENERGY (*units type*), 157
NX_FLOAT (*data type*), 156
NX_FLUX (*units type*), 157
NX_FREQUENCY (*units type*), 157
NX_INT (*data type*), 156
NX_LENGTH (*units type*), 158
NX_MASS (*units type*), 158
NX_MASS_DENSITY (*units type*), 158
NX_MOLECULAR_WEIGHT (*units type*), 158
NX_NUMBER (*data type*), 156
NX_PCOMPLEX (*data type*), 156
NX_PER_AREA (*units type*), 158
NX_PER_LENGTH (*units type*), 158
NX_PERIOD (*units type*), 158
NX_POSINT (*data type*), 156
NX_POWER (*units type*), 158
NX_PRESSURE (*units type*), 158
NX_PULSES (*units type*), 158
NX_QUATERNION (*data type*), 156
NX_SCATTERING_LENGTH_DENSITY (*units type*), 158
NX_SOLID_ANGLE (*units type*), 158
NX_TEMPERATURE (*units type*), 158
NX_TIME (*units type*), 158
NX_TIME_OF_FLIGHT (*units type*), 159
NX_TRANSFORMATION (*units type*), 159
NX_UINT (*data type*), 156
NX_UNITLESS (*units type*), 159
NX_VOLTAGE (*units type*), 159
NX_VOLUME (*units type*), 159
NX_WAVELENGTH (*units type*), 159
NX_WAVENUMBER (*units type*), 159
NXaberration (*base class*), 166
 used in application definition, 454
 used in base class, 201
NXactivity (*base class*), 168
 used in base class, 301
NXactuator (*base class*), 168
 used in application definition, 454
 used in base class, 256, 281, 302, 314, 317,
 327
NXaperture (*base class*), 170
 used in application definition, 432, 454,
 601, 942, 983, 1017, 1026
 used in base class, 201, 256, 302, 314, 338,
 394, 834, 883, 924
NXapm (*application definition*), 599
NXapm_composition_space_results (*application
definition*), 629
NXapm_input_ranging (*base class*), 638
 used in application definition, 629, 640,
 648, 660, 680, 686, 690, 698, 704, 709
NXapm_input_reconstruction (*base class*), 639
 used in application definition, 629, 640,
 648, 660, 680, 686, 690, 698, 704, 709
NXapm_paraprobe_config_clusterer (*application
definition*), 640
NXapm_paraprobe_config_distancer (*application
definition*), 648
NXapm_paraprobe_config_intersector (*application
definition*), 654
NXapm_paraprobe_config_nanochem (*application
definition*), 660
NXapm_paraprobe_config_ranger (*application
definition*), 679
NXapm_paraprobe_config_selector (*application
definition*), 686
NXapm_paraprobe_config_spatstat (*application
definition*), 689
NXapm_paraprobe_config_surfacer (*application
definition*), 697
NXapm_paraprobe_config_tessellator (*application
definition*), 704
NXapm_paraprobe_config_transcoder (*application
definition*), 709
NXapm_paraprobe_results_clusterer (*application
definition*), 711
NXapm_paraprobe_results_distancer (*application
definition*), 720
NXapm_paraprobe_results_intersector (*application
definition*), 728
NXapm_paraprobe_results_nanochem (*application*

definition), 736

NXapm_paraprobe_results_ranger (*application definition*), **776**

NXapm_paraprobe_results_selector (*application definition*), **785**

NXapm_paraprobe_results_spatstat (*application definition*), **791**

NXapm_paraprobe_results_surfacer (*application definition*), **799**

NXapm_paraprobe_results_tessellator (*application definition*), **810**

NXapm_paraprobe_results_transcoder (*application definition*), **824**

NXarchive (*application definition*), **424**

NXarpes (*application definition*), **428**

NXatom (*base class*), **172**
 used in application definition, **454**
 used in base class, **189, 271, 352, 415**

NXattenuator (*base class*), **174**
 used in application definition, **515, 584, 1017, 1026, 1043**
 used in base class, **314, 834**

NXbeam (*base class*), **175**
 used in application definition, **454, 515, 601, 897, 942, 1051**
 used in base class, **256, 302, 314, 378, 884, 1001**

NXbeam_path (*base class*), **834**
 used in application definition, **919, 983**

NXbeam_splitter (*base class*), **841**
 used in base class, **834**

NXbeam_stop (*base class*), **181**
 used in base class, **314**

NXBending_magnet (*base class*), **182**
 used in base class, **314**

nxbrowse, **15**

nxbrowse (*utility*), **1081**

NXcalibration (*base class*), **184**
 used in application definition, **942**
 used in base class, **190, 371**

NXcanSAS (*application definition*), **431**

NXcanSAS (*applications*)
 dQ1, **440**
 dQw, **440**
 I, **437**
 Idev, **439**
 Q, **435**
 Qdev, **439**
 resolutions, **436**
 SASaperture, **441**
 SAScollimation, **442**
 SASdata, **434**
 SASdetector, **442**
 SASentry, **433**

SASinstrument, **441**

SASnote, **446**

SASprocess, **445**

SASprocessnote, **446**

SASSample, **444**

SASsource, **443**

SAStransmission_spectrum, **446**

Tdev, **447**

NXcapillary (*base class*), **187**
 used in base class, **314**

NXcg_alpha_complex (*base class*), **845**
 used in application definition, **800**

NXcg_cylinder_set (*base class*), **847**
 used in application definition, **640, 648, 660, 680, 686, 690, 698, 704**
 used in base class, **873, 1037**

NXcg_ellipsoid_set (*base class*), **849**
 used in application definition, **640, 648, 660, 680, 686, 690, 698, 704**
 used in base class, **873, 1037**

NXcg_face_list_data_structure (*base class*), **850**
 used in application definition, **640, 648, 660, 680, 686, 690, 698, 704, 737, 800, 810**
 used in base class, **859, 863, 867, 869, 876, 878**

NXcg_geodesic_mesh (*base class*), **853**

NXcg_grid (*base class*), **854**
 used in application definition, **629, 737, 949, 968**
 used in base class, **862, 931**

NXcg_half_edge_data_structure (*base class*), **856**
 used in base class, **859, 869, 876, 879**

NXcg_hexahedron_set (*base class*), **858**
 used in application definition, **640, 648, 660, 680, 686, 690, 698, 704, 737**
 used in base class, **867, 878, 1037**

NXcg_marching_cubes (*base class*), **862**
 used in application definition, **737**

NXcg_parallelogram_set (*base class*), **863**

NXcg_point_set (*base class*), **865**
 used in application definition, **949**
 used in base class, **846, 958**

NXcg_polygon_set (*base class*), **866**

NXcg_polyhedron_set (*base class*), **869**
 used in application definition, **737, 810, 949, 968**
 used in base class, **855, 873, 958**

NXcg_polyline_set (*base class*), **871**
 used in base class, **958**

NXcg_roi_set (*base class*), **873**
 used in application definition, **949, 968**

NXcg_sphere_set (*base class*), **874**
 used in base class, **873**

NXcg_tetrahedron_set (*base class*), **875**

used in application definition, 800
 used in base class, 846

NXcg_triangle_set (base class), 877
 used in application definition, 737, 800
 used in base class, 846, 879, 958

NXcg_triangulated_surface_mesh (base class), 879
 used in base class, 854

NXcg_unit_normal_set (base class), 880
 used in application definition, 737
 used in base class, 859, 863, 867, 869, 871, 876, 878

NXchamber (base class), 881
 used in application definition, 601

NXchemical_composition (base class), 188
 used in application definition, 601, 737

NXcircuit (base class), 190
 used in base class, 210, 211, 215, 388

NXcite (base class), 192
 used in application definition, 454, 908

NXclass (attribute), 38

NXclustering (base class), 881

NXcollection (base class), 193
 used in application definition, 432, 454, 515, 544, 601, 654, 737, 897, 961, 1017, 1026
 used in base class, 235, 262, 277, 314, 341, 408, 1001

NXcollectioncolumn (base class), 883
 used in application definition, 942
 used in base class, 915

NXcollimator (base class), 193
 used in application definition, 432, 535, 539
 used in base class, 314

NXcomponent (base class), 195
 used in application definition, 454
 used in base class, 256, 302, 317

NXcontainer (base class), 884

nxconvert (utility), 1081

NXcoordinate_system (base class), 197
 used in application definition, 454
 used in base class, 262, 888

NXcoordinate_system_set (base class), 887
 used in application definition, 601, 629, 711, 720, 728, 737, 776, 785, 792, 800, 810, 824, 949, 968

NXcorrector_cs (base class), 200
 used in application definition, 454
 used in base class, 256

NXcrystal (base class), 204
 used in application definition, 513, 551, 1017, 1026
 used in base class, 314, 338

NXcs_computer (base class), 209
 used in application definition, 629, 711, 720, 728, 737, 776, 785, 792, 800, 810, 824, 968
 used in base class, 212

NXcs_cpu (base class), 891
 used in application definition, 629, 711, 720, 728, 737, 776, 785, 792, 800, 810, 824, 968

NXcs_filter_boolean_mask (base class), 891
 used in application definition, 640, 648, 660, 680, 690, 698, 704, 711, 720, 737, 776, 785, 792, 800, 810
 used in base class, 1037

NXcs_gpu (base class), 892
 used in application definition, 629, 711, 720, 728, 737, 776, 785, 792, 800, 810, 824, 968

NXcs_io_obj (base class), 893
 used in application definition, 629, 711, 720, 728, 737, 776, 785, 792, 800, 810, 824, 968
 used in base class, 894

NXcs_io_sys (base class), 894
 used in application definition, 629, 711, 720, 728, 737, 776, 785, 792, 800, 810, 824, 968

NXcs_memory (base class), 210
 used in base class, 209

NXcs_mm_sys (base class), 894
 used in application definition, 629, 711, 720, 728, 737, 776, 785, 792, 800, 810, 824, 968

NXcs_prng (base class), 895
 used in application definition, 690

NXcs_process (base class), 211
 used in base class, 209

NXcs_profiling (base class), 212
 used in application definition, 454, 629, 648, 654, 660, 680, 690, 698, 704, 709, 711, 720, 728, 737, 776, 785, 792, 800, 810, 824, 961, 968

NXcs_profiling_event (base class), 214
 used in application definition, 629, 711, 720, 728, 737, 776, 785, 792, 800, 810, 824, 968
 used in base class, 212

NXcs_storage (base class), 215
 used in base class, 209

NXcsg (base class), 896
 used in application definition, 601
 used in base class, 896, 1036

NXcxi_ptycho (application definition), 897

NXcylindrical_geometry (base class), 216
 used in base class, 181, 235

NXdata (base class), 5, 54, 217

used in application definition, 428, 432, 454, 505, 508, 511, 513, 515, 530, 532, 535, 539, 542, 544, 546, 548, 551, 555, 557, 560, 562, 565, 568, 570, 573, 574, 577, 579, 580, 582, 584, 601, 737, 897, 908, 932, 942, 949, 968, 983, 1009, 1017, 1026, 1043, 1051

used in base class, 176, 183, 184, 188, 204, 235, 262, 271, 276, 277, 290, 296, 299, 305, 312, 321, 331, 333, 338, 341, 351, 371, 372, 378, 385, 394, 400, 408, 841, 998, 1007

NXdeflector (*base class*), **234**

- used in application definition, 454
- used in base class, 201, 256, 302, 388, 394, 883, 915, 924, 1039

NXdelocalization (*base class*), **902**

- used in application definition, 737

NXdetector (*base class*), **234**

- used in application definition, 428, 432, 454, 505, 511, 513, 515, 530, 532, 535, 539, 542, 548, 551, 555, 557, 560, 562, 565, 570, 574, 577, 579, 581, 582, 584, 601, 897, 919, 942, 1017, 1026, 1043, 1051
- used in base class, 314, 317, 915

NXdetector_channel (*base class*), **246**

- used in application definition, 515
- used in base class, 235

NXdetector_group (*base class*), **248**

- used in application definition, 515
- used in base class, 314

NXdetector_module (*base class*), **250**

- used in application definition, 515
- used in base class, 235

nxdir (*utility*), **1081**

NXdirectttof (*application definition*), **450**

NXdisk_chopper (*base class*), **252**

- used in application definition, 450, 532, 1017, 1026
- used in base class, 314, 834

NXdispersion (*base class*), **903**

- used in application definition, 908

NXdispersion_function (*base class*), **904**

- used in application definition, 908
- used in base class, 903

NXdispersion_repeated_parameter (*base class*), **906**

- used in application definition, 908
- used in base class, 904

NXdispersion_single_parameter (*base class*), **906**

- used in application definition, 908
- used in base class, 904

NXdispersion_table (*base class*), **907**

- used in application definition, 908
- used in base class, 903

NXdispersive_material (*application definition*), **908**

NXdistortion (*base class*), **255**

NXDL, **22**, **74**, **131**, **134**, **134**

NXDL elements, **135**

NXDL template file, **65**

nxdl_to_hdf5.py, **1082**

NXbeam_column (*base class*), **256**

- used in application definition, 454
- used in base class, 317

NXelectron_detector (*base class*), **260**

NXelectronanalyser (*base class*), **915**

- used in application definition, 942

NXelectrostatic_kicker (*base class*), **917**

NXellipsometry (*application definition*), **918**

NXem (*application definition*), **451**

NXem_ebsd (*base class*), **261**

- used in application definition, 454
- used in base class, 372

NXem_eds (*base class*), **270**

- used in application definition, 454
- used in base class, 372

NXem_eels (*base class*), **273**

- used in application definition, 454
- used in base class, 372

NXem_img (*base class*), **274**

- used in application definition, 454
- used in base class, 372

NXem_measurement (*base class*), **275**

- used in application definition, 454

NXem_simulation (*base class*), **276**

- used in application definition, 454

NXenergydispersion (*base class*), **923**

- used in application definition, 942
- used in base class, 915

NXentry (*base class*), **5**, **276**

- used in application definition, 425, 428, 432, 450, 454, 505, 507, 508, 511, 513, 515, 530, 532, 535, 539, 542, 544, 546, 548, 551, 555, 557, 560, 562, 565, 568, 570, 573, 574, 577, 579–582, 584, 601, 629, 640, 648, 654, 660, 680, 686, 690, 698, 704, 709, 711, 720, 728, 737, 776, 785, 792, 800, 810, 824, 897, 908, 919, 932, 933, 936, 942, 949, 961, 968, 983, 1009, 1017, 1026, 1043, 1051
- used in base class, 373

NXenvironment (*base class*), **281**

- used in application definition, 932, 983, 1009
- used in base class, 378

NXevent_data (*base class*), **282**

- used in application definition, 1017
- used in base class, 314

NXevent_data_em (*base class*), **284**

- used in application definition, 454
- used in base class, 275

NXfabrication (*base class*), [287](#)
 used in application definition, [454](#), [601](#),
 [629](#), [711](#), [720](#), [728](#), [737](#), [776](#), [785](#), [792](#), [800](#),
 [810](#), [824](#), [933](#), [936](#), [968](#), [1043](#)
 used in base class, [169](#), [190](#), [195](#), [256](#), [314](#),
 [317](#), [389](#), [394](#), [891](#), [893](#), [1001](#)

NXfermi_chopper (*base class*), [288](#)
 used in application definition, [450](#), [544](#),
 [1026](#)
 used in base class, [314](#)

NXfiber (*base class*), [925](#)
 used in base class, [834](#)

NXfilter (*base class*), [289](#)
 used in base class, [314](#), [834](#)

NXflipper (*base class*), [293](#)
 used in base class, [314](#)

NXfluo (*application definition*), [505](#)

NXfresnel_zone_plate (*base class*), [294](#)

NXgeometry (*base class*), [295](#)
 used in application definition, [535](#), [539](#),
 [1017](#), [1026](#)
 used in base class, [170](#), [181](#), [183](#), [194](#), [204](#),
 [235](#), [252](#), [281](#), [288](#), [290](#), [299](#), [312](#), [331](#), [333](#),
 [335](#), [338](#), [347](#), [378](#), [389](#), [394](#), [415](#), [419](#), [834](#)

NXgraph_edge_set (*base class*), [928](#)
 used in base class, [930](#)

NXgraph_node_set (*base class*), [929](#)
 used in base class, [930](#)

NXgraph_root (*base class*), [930](#)

NXgrating (*base class*), [296](#)
 used in application definition, [1043](#)
 used in base class, [338](#), [834](#)

NXguide (*base class*), [298](#)
 used in base class, [314](#)

NXhistory (*base class*), [301](#)
 used in base class, [314](#), [378](#), [385](#)

NXibeam_column (*base class*), [302](#)
 used in application definition, [454](#)
 used in base class, [317](#)

NXimage (*base class*), [304](#)
 used in application definition, [454](#)
 used in base class, [201](#), [271](#), [275](#), [285](#), [372](#)

NXindirecttof (*application definition*), [507](#)

nxitgest (*utility*), [1081](#)

NXinsertion_device (*base class*), [312](#)
 used in base class, [314](#)

NXinstrument (*base class*), [6](#), [314](#)
 used in application definition, [425](#), [428](#),
 [432](#), [450](#), [505](#), [507](#), [508](#), [511](#), [513](#), [515](#), [530](#),
 [532](#), [535](#), [539](#), [542](#), [544](#), [546](#), [548](#), [551](#), [555](#),
 [557](#), [560](#), [562](#), [565](#), [568](#), [570](#), [574](#), [577](#), [579](#)–
 [582](#), [584](#), [601](#), [824](#), [897](#), [919](#), [932](#), [942](#), [983](#),
 [1009](#), [1017](#), [1026](#), [1043](#), [1051](#)
 used in base class, [277](#), [408](#)

NXinstrument_em (*base class*), [316](#)
 used in application definition, [454](#)
 used in base class, [275](#), [285](#)

NXinteraction_volume_em (*base class*), [320](#)
 used in application definition, [454](#)

NXion (*base class*), [321](#)
 used in application definition, [454](#), [601](#),
 [629](#), [737](#), [776](#), [824](#)
 used in base class, [302](#), [415](#)

NXiqproc (*application definition*), [508](#)

NXisocontour (*base class*), [931](#)
 used in application definition, [737](#)

NXiv_temp (*application definition*), [932](#)

NXlab_electro_chemo_mechanical_preparation
 (*application definition*), [933](#)

NXlab_sample_mounting (*application definition*), [936](#)

NXlauetof (*application definition*), [510](#)

NXLens_em (*base class*), [323](#)
 used in application definition, [454](#), [601](#)
 used in base class, [201](#), [256](#), [302](#), [394](#), [883](#),
 [915](#), [924](#), [1039](#)

NXLens_opt (*base class*), [937](#)
 used in application definition, [919](#)
 used in base class, [834](#)

NXlog (*base class*), [324](#)
 used in application definition, [1017](#), [1026](#)
 used in base class, [194](#), [204](#), [290](#), [327](#), [333](#),
 [335](#), [341](#), [353](#), [378](#), [389](#), [917](#), [940](#), [1005](#), [1013](#),
 [1035](#), [1038](#)

NXmagnetic_kicker (*base class*), [940](#)

NXmanipulator (*base class*), [327](#)
 used in application definition, [454](#), [601](#),
 [942](#)
 used in base class, [317](#)

NXmatch_filter (*base class*), [941](#)
 used in application definition, [640](#), [648](#),
 [660](#), [680](#), [686](#), [690](#), [698](#), [704](#), [737](#)

NXmirror (*base class*), [331](#)
 used in base class, [314](#), [834](#)

NXmoderator (*base class*), [333](#)
 used in application definition, [1017](#), [1026](#)
 used in base class, [314](#)

NXmonitor (*base class*), [335](#)
 used in application definition, [505](#), [511](#),
 [513](#), [530](#), [532](#), [535](#), [539](#), [542](#), [548](#), [551](#), [555](#),
 [557](#), [560](#), [562](#), [565](#), [570](#), [574](#), [601](#), [897](#), [1017](#),
 [1026](#)
 used in base class, [277](#), [408](#)

NXmonochromator (*base class*), [337](#)
 used in application definition, [428](#), [454](#),
 [505](#), [507](#), [530](#), [535](#), [548](#), [570](#), [574](#), [919](#), [1043](#)
 used in base class, [256](#), [302](#), [314](#), [834](#)

NXmonopd (*application definition*), [513](#)

NXmpes (*application definition*), [941](#)

NXms (*application definition*), 949
NXms_feature_set (*base class*), 958
 used in application definition, 949
NXms_score_config (*application definition*), 960
NXms_score_results (*application definition*), 968
NXms_snapshot (*base class*), 981
 used in application definition, 949, 968
 used in base class, 982
NXms_snapshot_set (*base class*), 981
 used in application definition, 949, 968
NXmx (*application definition*), 515
NXnote (*base class*), 339
 used in application definition, 432, 454,
 601, 942, 1017, 1026, 1051
 used in base class, 168, 170, 184, 235, 262,
 277, 281, 301, 305, 341, 358, 371, 394, 400,
 408, 421, 1041
NXObject (*base class*), 341
 used in application definition, 949, 968
NXoff_geometry (*base class*), 344
 used in base class, 170, 174, 181, 183, 194,
 204, 235, 252, 288, 290, 296, 299, 312, 331,
 333, 335, 338, 378, 389, 394, 419, 421, 1036
NXopt (*application definition*), 982
NXoptical_system_em (*base class*), 345
 used in application definition, 454
 used in base class, 317
NXorientation (*base class*), 347
 used in application definition, 1017, 1026
 used in base class, 296, 389
NXorientation_set (*base class*), 996
 used in application definition, 949
 used in base class, 859, 863
NXparameters (*base class*), 348
 used in application definition, 454, 508,
 546, 568, 573
 used in base class, 184, 276, 277, 341, 371,
 408
NXpdb (*base class*), 348
NXpeak (*base class*), 351
 used in application definition, 601
 used in base class, 271, 274
NXphase (*base class*), 352
 used in application definition, 454
 used in base class, 262
NXpid_controller (*base class*), 352
 used in application definition, 1009
 used in base class, 169, 327
NXpinhole (*base class*), 355
 used in base class, 834
NXplot (*utility*), 1082
NXpolarizer (*base class*), 356
 used in application definition, 1017, 1026
 used in base class, 314
NXpolarizer_opt (*base class*), 997
 used in base class, 834
NXpositioner (*base class*), 356
 used in application definition, 1017, 1026,
 1051
 used in base class, 170, 314, 327, 378
NXprocess, 69
NXprocess (*base class*), 358
 used in application definition, 432, 454,
 508, 546, 568, 573, 601, 629, 640, 648, 654,
 660, 680, 686, 690, 698, 704, 709, 711, 720,
 728, 737, 776, 785, 792, 800, 810, 824, 919,
 933, 942, 949, 961, 968, 983, 1009, 1051
 used in base class, 201, 262, 271, 274, 276,
 277, 305, 321, 372, 400, 408, 958, 1014
NXprogram (*base class*), 359
 used in application definition, 454, 601,
 629, 949, 961, 968, 983
 used in base class, 195, 271, 274, 276, 305,
 400
NXpulser_apm (*base class*), 1001
 used in application definition, 601
NXpump (*base class*), 360
 used in application definition, 454, 601
 used in base class, 317
NXquadric (*base class*), 1003
 used in base class, 1036
NXquadrupole_magnet (*base class*), 1004
NXreflections (*base class*), 360
NXreflectron (*base class*), 1005
 used in application definition, 601
NXrefscan (*application definition*), 530
NXreftof (*application definition*), 532
NXregion (*base class*), 1006
NXregistration (*base class*), 370
NXresolution (*base class*), 370
NXroi (*base class*), 372
 used in application definition, 454
NXroot (*base class*), 373
 attributes, 19
NXrotations (*base class*), 375
 used in base class, 262
NXsample (*base class*), 5, 378
 used in application definition, 425, 428,
 432, 454, 505, 508, 511, 513, 515, 530, 532,
 535, 539, 542, 544, 546, 548, 551, 555, 557,
 560, 562, 565, 568, 570, 573, 574, 577, 579,
 582, 584, 601, 897, 908, 919, 933, 936, 942,
 949, 968, 983, 1009, 1017, 1026, 1043, 1051
 used in base class, 277, 408, 841, 925, 937,
 998, 1049
NXsample_component (*base class*), 385
 used in base class, 378
NXsas, 1087

- N**
- NXsas (*application definition*), 534**
 - NXsastof (*application definition*), 538**
 - NXscan (*application definition*), 542**
 - NXscanbox_em (*base class*), 387**
 - used in application definition, 454
 - used in base class, 256, 302
 - NXsensor (*base class*), 389**
 - used in application definition, 454, 932, 983, 1009
 - used in base class, 256, 281, 290, 302, 314, 317, 327, 353
 - NXsensor_scan (*application definition*), 1009**
 - NXseparator (*base class*), 1013**
 - NXshape (*base class*), 392**
 - used in application definition, 535, 539, 1017, 1026
 - used in base class, 204, 296, 331, 841, 884, 998
 - NXsimilarity_grouping (*base class*), 1014**
 - used in application definition, 711
 - NXslip_system_set (*base class*), 1016**
 - NXslit (*base class*), 393**
 - used in application definition, 1043
 - used in base class, 834
 - NXsnsevent (*application definition*), 1017**
 - NXsnshisto (*application definition*), 1026**
 - NXsolenoid_magnet (*base class*), 1035**
 - NXsolid_geometry (*base class*), 1036**
 - NXsource (*base class*), 394**
 - used in application definition, 425, 428, 432, 454, 505, 508, 513, 515, 530, 535, 539, 546, 548, 551, 562, 565, 568, 570, 574, 580, 601, 897, 919, 942, 1017, 1026, 1043
 - used in base class, 256, 302, 314, 834, 1001
 - NXspatial_filter (*base class*), 1036**
 - used in application definition, 640, 648, 660, 680, 686, 690, 698, 704
 - NXspe (*application definition*), 543**
 - NXspectrum (*base class*), 400**
 - used in application definition, 454
 - used in base class, 274, 285, 372
 - NXspin_rotator (*base class*), 1037**
 - NXspindispersion (*base class*), 1039**
 - used in base class, 915
 - NXsqom (*application definition*), 545**
 - NXstxm (*application definition*), 547**
 - NXsubentry (*base class*), 407**
 - used in application definition, 983
 - used in base class, 277
 - NXsubsampling_filter (*base class*), 1040**
 - used in application definition, 640, 648, 660, 680, 686, 690, 698, 704
 - NXsubstance (*base class*), 1041**
 - nxsummary, 1081**
 - NXtas (*application definition*), 551**
 - NXtofnpd (*application definition*), 554**
 - NXtofraw (*application definition*), 557**
 - NXtofsingle (*application definition*), 559**
 - NXtomo (*application definition*), 562**
 - NXtomophase (*application definition*), 565**
 - NXtomoproc (*application definition*), 568**
 - NXtransformations (*base class*), 411**
 - used in application definition, 515, 601, 629, 711, 720, 728, 737, 776, 785, 792, 800, 810, 824, 897, 949, 968
 - used in base class, 169, 176, 195, 198, 281, 296, 314, 370, 834, 847, 849, 854, 855, 859, 863, 866, 869, 871, 874, 876, 878, 884, 996, 1001, 1006
 - nxtranslate (*utility*), 1082**
 - NXtranslation (*base class*), 414**
 - used in application definition, 1017, 1026
 - used in base class, 296
 - NXtransmission (*application definition*), 1043**
 - NXunit_cell (*base class*), 415**
 - used in application definition, 454
 - used in base class, 352
 - NXuser (*base class*), 418**
 - used in application definition, 425, 454, 555, 557, 560, 601, 629, 711, 720, 728, 737, 776, 785, 792, 800, 810, 824, 933, 936, 942, 949, 968, 983, 1009, 1017, 1026, 1043
 - used in base class, 277, 285, 408
 - nxvalidate, 73**
 - NXvelocity_selector (*base class*), 419**
 - used in base class, 314, 338
 - NXwaveplate (*base class*), 1048**
 - used in application definition, 919
 - used in base class, 834
 - NXxas (*application definition*), 570**
 - NXxasproc (*application definition*), 572**
 - NXxbase (*application definition*), 574**
 - NXxeuler (*application definition*), 577**
 - NXxkappa (*application definition*), 578**
 - NXxlaue (*application definition*), 580**
 - NXxlaueplate (*application definition*), 581**
 - NXxbnb (*application definition*), 582**
 - NXxpccs (*application definition*), 1051**
 - NXxraylens (*base class*), 421**
 - used in base class, 314, 834
 - NXxrot (*application definition*), 583**
- O**
- object (*base class*), see NXobject, 341**
 - objects (*field*), 882, 902, 996**
 - observed_frame (*field*), 363**
 - observed_frame_errors (*field*), 363**
 - observed_frame_var (*field*), 363**

observed_phi (field), 364
observed_phi_errors (field), 364
observed_phi_var (field), 364
observed_px_x (field), 363
observed_px_x_errors (field), 364
observed_px_x_var (field), 363
observed_px_y (field), 364
observed_px_y_errors (field), 364
observed_px_y_var (field), 364
observed_x (field), 364
observed_x_errors (field), 365
observed_x_var (field), 365
observed_y (field), 365
observed_y_errors (field), 365
observed_y_var (field), 365
occupancy (field), 173
odd_layer_density (field), 332
odd_layer_material (field), 332
off_geometry (base class), *see NXoff_geometry*, 344
offset (field attribute), 26, 179, 250, 251, 283, 413, 524, 569
offset (field), 186, 191, 232, 802, 846
offset_units (field attribute), 250, 251, 413
offset_values (field), 701
OpenGENIE (data analysis software), 1083
operating_frequency (field), 190
operating_system (field), 210, 633, 715, 724, 732, 760, 780, 788, 795, 805, 818, 829, 975
operation (field), 896
operation_mode (field), 256, 302, 476, 479, 602
opt (application definition), *see NXopt*, 982
optical_loss (field), 842
optical_system_em (base class), *see NXoptical_system_em*, 345
ORCID (field), 418
orcid (field), 603, 631, 712, 721, 729, 738, 777, 786, 793, 801, 812, 825, 942, 951, 970, 984, 1010
orcid_platform (field), 603, 631, 712, 721, 729, 738, 777, 786, 793, 801, 812, 825, 951, 970
order (transformation), *see depends on (field attribute)*
order_no (field), 205
orientation (base class), *see NXorientation*, 347
orientation (field), 643, 649, 661, 681, 692, 699, 705, 748, 757, 759, 760, 850, 880, 954, 997
orientation_angle (field), 837, 989
orientation_euler (field), 377
orientation_matrix (field), 206, 291, 380, 386, 511, 552, 575
orientation_model (field), 962
orientation_parameterization_sign_convention (field), 951, 971
orientation_quaternion (field), 377
orientation_set (base class), *see NXorientation_set*, 996
origin (field), 198, 262, 459–461, 463, 481, 631, 741, 855, 952, 971
original_axis (field), 185
original_centre (field), 255
original_points (field), 255
Osborn, Raymond, 1089
other_material (field), 837, 989
other_shape (field), 843, 999
other_type (field), 842, 938, 1049
outer_diameter (field), 294
outermost_zone_width (field), 294
output_channels (field), 191
output_heater_power (field), 328
output_impedance (field), 190
output_signal (field), 191
output_slew_rate (field), 191
outputs (field), 196
outputVALUE (field), 169
overlaps (field), 367

P

packing_fraction (field), 886
pair_separation (field), 253
parameter (field), 617
parameter_type (field), 986
parameter_type_name (field), 987
parameter_units (field), 905, 906
parameterization (field), 954, 996
parameters (base class), *see NXparameters*, 348
parameters (field), 1003
parent (field), 1007
parent_identifier (field), 959
parent_mask (field), 1007
partiality (field), 362
pass_energy (field), 429, 924, 944
path_length (field), 381
path_length_window (field), 382
pdb (base class), *see NXpdb*, 348
peak (base class), *see NXpeak*, 351
peak (field), 272
peak_model (field), 620
peak_power (field), 397, 836
period (field), 297, 396
phase (base class), *see NXphase*, 352
phase (field), 253, 313
phase_id (field), 352
phases_per_scan_point (field), 266
phi (field), 578, 579, 666
physical_file_format, *see file format*
physical_form (field), 382, 457
physical_quantity (field), 169, 185, 371, 480
PhysicsCgms, 596
pid_controller (base class), *see NXpid_controller*, 352

pinhole (*base class*), *see* NXpinhole, 355
pitch (*field*), 442, 445
pixel_id (*field*), 1020, 1028
pixel_mask (*field*), 240, 248, 520
pixel_mask_applied (*field*), 239, 248, 520
plotting, 4, 5, 12, 14, 18, 19, 22, 23, 23, 33, 54, 74, 185, 227, 229–231, 277, 341, 371, 374, 389, 408, 433, 1082
 how to find data, 46
point_group (*field*), 381, 387, 416
point_normal_form (*field*), 756
poison_depth (*field*), 334
poison_material (*field*), 334
polar (*field*), 544
polar_angle (*field*), 207, 237, 367, 508, 511, 514, 531, 533, 536, 540, 552, 555, 558, 560, 577, 579, 582, 584, 1020, 1029
polar_width (*field*), 544
polarizer (*base class*), *see* NXpolarizer, 356
polarizer (*field*), 1045
polarizer_angle (*field*), 998
polarizer_opt (*base class*), *see* NXpolarizer_opt, 997
polarizing (*field*), 842
poles (*field*), 313
polylines (*field*), 872
populated_elements (*field attribute*), 1054
position (*field*), 173, 319, 351, 480, 632, 855, 857, 866
positioner (*base class*), *see* NXpositioner, 356
power (*field*), 313, 396, 610, 1002
power_consumption (*field*), 191
power_loss (*field*), 926
power_setting (*field*), 323, 476, 479
power_source (*field*), 190
pre_factor (*field*), 964
pre_sample_flightpath (*field*), 278, 409, 555, 557, 560
precompiled_executable, *see* NAPI installation
predicted_frame (*field*), 362
predicted_phi (*field*), 363
predicted_px_x (*field*), 363
predicted_px_y (*field*), 363
predicted_x (*field*), 362
predicted_y (*field*), 363
preparation_date (*field*), 381, 427, 457, 606, 946, 990
preprocessing_kernel_width (*field*), 700
preprocessing_method (*field*), 700
preset (*field*), 335, 506, 512, 514, 531, 533, 537, 540, 553, 556, 558, 561, 571, 576
pressure (*field*), 379, 427, 611
previous_source (*field*), 397
prf_cc (*field*), 366
primary (*field attribute*), 235–237
probability_mass (*field*), 794, 795
probe (*field*), 257, 346, 395, 426, 429, 465, 506, 509, 513, 530, 535, 539, 546, 548, 552, 563, 566, 568, 570, 574, 898, 943, 1019, 1028
probe_current (*field*), 346
procedure (*field*), 989
process (*base class*), *see* NXprocess, 358
process_identifier (*field*), 816
Processed Data, 69
profile (*field*), 526
program (*base class*), *see* NXprogram, 359
program (*field*), 281, 358, 359, 425, 455, 464, 480, 509, 546, 568, 573, 602, 614–620, 630, 640, 648, 654, 660, 680, 686, 690, 698, 704, 709, 711, 720, 729, 737, 776, 785, 792, 800, 811, 824, 862, 895, 909, 921, 985, 992, 993, 1010
program_name (*field*), 278, 409, 544, 950, 961, 970
program_url (*field attribute*), 1010
programs, 1081
projection (*field*), 883, 944
protection_features (*field*), 191
protocol_name (*field*), 617
proton_charge (*field*), 1017, 1026
psi (*field*), 544
pub_chem_link (*field attribute*), 1042
pulse_delay (*field*), 178
pulse_duration (*field*), 178
pulse_energy (*field*), 178, 397, 610, 1002
pulse_fraction (*field*), 610, 1001
pulse_frequency (*field*), 610, 1001
pulse_height (*field*), 283
pulse_id (*field*), 615
pulse_mode (*field*), 610, 1001
pulse_number (*field*), 1002
pulse_time (*field*), 1020
pulse_width (*field*), 397
pulsed_voltage (*field*), 610, 1001
pulser_apm (*base class*), *see* NXpulser_apm, 1001
pulses_since_last_ion (*field*), 615
pump (*base class*), *see* NXpump, 360
punx (*utility*), 1082
PyMCA (*data analysis software*), 1084

Q

Q (*field*), 435
Q_indices (*group attribute*), 434
Qdev (*field*), 439
qh (*field*), 552
qk (*field*), 552
ql (*field*), 552
Qmean (*field*), 440
quadric (*base class*), *see* NXquadric, 1003
quadrupole_magnet (*base class*), *see* NXquadrupole_magnet, 1004
qx (*field*), 509, 546
qy (*field*), 509, 546

`qz (field)`, 547

R

`r_slit (field)`, 288

`radiation (field)`, 443

`radii (field)`, 643, 650, 662, 681, 687, 692, 699, 705, 848, 875

`radius (field)`, 253, 288, 419, 874, 964

`randomize_ion_types (field)`, 690

`range (field)`, 336

`range_increment (field)`, 619

`range_minmax (field)`, 619

`rank`, 14, 50, 55, 66

`rank (NXDL attribute)`, 66

`ratio (field)`, 253

`raw (field)`, 945

`raw_file (field)`, 569, 573

`raw_frames (field)`, 1017, 1026

`raw_time_of_flight (field)`, 235

`raw_tof (field)`, 616

`raw_value (field)`, 325, 357

`rdeform_field (field)`, 255

`read_file`, 15

`real (field)`, 306–309, 469–472

`real_time (field)`, 238

`realloc_cell_cache (field)`, 965

`reconstructed_positions (field)`, 617, 828

`recover_evaporation_id (field)`, 641

`recrystallized_grain_identifier (field)`, 973

`recrystallized_volume_fraction (field)`, 973

`rediscretization (field)`, 965

`reference_beam (field attribute)`, 178

`reference_burgers_magnitude (field)`, 963

`reference_data_link (field)`, 837, 989, 992

`reference_frame (field attribute)`, 173

`reference_frame (field)`, 376, 415

`reference_frame_type (field)`, 952, 971

`reference_plane (field)`, 179

`reference_shear_modulus (field)`, 963

`reflectance (field)`, 842, 938, 1049

`reflection (field)`, 206, 356, 998

`reflection_id (field)`, 361

`reflections (base class)`, see `NXreflections`, 360

`reflectivity (field)`, 993

`reflectron (base class)`, see `NXreflectron`, 1005

`refractive_index (field)`, 908, 910, 911

`refscan (application definition)`, see `NXrefscan`, 530

`reftof (application definition)`, see `NXreftof`, 532

`region (base class)`, see `NXregion`, 1006

`region_origin (field)`, 430

`region_size (field)`, 430

`region_type (file attribute)`, 1007

`registration (base class)`, see `NXregistration`, 370

`regular expression`, 38

`relative_intensity (field)`, 268

`relative_molecular_mass (field)`, 380, 386, 886

`relative_resolution (field)`, 371

`relative_resolution_errors (field)`, 371

`release`

`NeXus definitions`, 1078

`notes`, 1078

`process`, 1078

`tags`, 131, 1078

`versioning`, 131, 1078

`release_date (field)`, 425

`removal (field)`, 935

`repository`, 1075, see `NAPI installation`

`representation (field)`, 905, 910–912

`reserved prefixes`, 40

`BLUESKY_`, 40

`DECTRIS_`, 40

`identifier`, 40

`IDF_`, 40

`NDAttr`, 40

`NX`, 40

`NX_`, 40

`PDBX_`, 40

`SAS_`, 40

`SILX_`, 40

`reserved suffixes`, 41

`end`, 41

`errors`, 41

`increment_set`, 41

`indices`, 41

`mask`, 41

`offset`, 41

`scaling_factor`, 41

`set`, 41

`weights`, 41

`resolution (base class)`, see `NXresolution`, 370

`resolution (field)`, 371

`resolution_errors (field)`, 371

`resolution_formula_description (field)`, 371

`resolutions (field attribute)`, 436

`resolutions_description (field attribute)`, 437

`response_time (field)`, 1046

`results_path (field)`, 630, 649, 654, 661, 680, 690, 698,

704, 710, 712, 721, 729, 738, 777, 793, 800,

811, 825, 961, 969

`retardance (field)`, 1049

`revision (field)`, 278, 409, 425

`revision history`, 1092

`revolutions (field)`, 921

`Riedel, Richard`, 1088

`roi (base class)`, see `NXroi`, 372

`roi_cylinder_height (field)`, 674

`roi_cylinder_radius (field)`, 674

`roi_identifier (field)`, 759

role (*field*), 418, 426, 456, 603, 631, 712, 721, 730, 738, 778, 787, 793, 801, 812, 825, 951, 970, 1022, 1031
roll (*field*), 442, 445
root (*base class*), *see* NXroot, 373
rotating_element_type (*field*), 921
rotation, 26
rotation (*field*), 319, 346, 480, 934
rotation_angle (*field*), 382, 514, 531, 533, 536, 540, 543, 544, 549, 552, 563, 566, 578, 579, 583, 584
rotation_angle_step (*field*), 584
rotation_control (*field*), 934
rotation_convention (*field*), 458, 889, 951, 971
rotation_euler (*field*), 376
rotation_handedness (*field*), 458, 888
rotation_quaternion (*field*), 376
rotation_speed (*field*), 253, 288, 419, 451
rotations (*base class*), *see* NXrotations, 375
RPM, *see* NAPI installation
rules, 3
 HDF, 18, 39, 133
 naming, 22, 30, 38, 133
 NeXus, 30
 NX prefix, 22
 XML, 133
run (*field*), 433
run_control (*field*), 390, 1011
run_cycle (*field*), 278, 409, 425
run_number (*field*), 557, 602, 1017, 1026

S

sacrifice_isotopic_uniqueness (*field*), 829
sample (*base class*), *see* NXsample, 378
sample_bias (*field*), 945
sample_component (*base class*), *see* NXsample_component, 385
sample_component (*field*), 381
sample_history (*field*), 606, 990
sample_id (*field*), 426
sample_name (*field*), 989
sample_orientation (*field*), 380, 386, 990
sample_symmetry (*field*), 376
sample_temperature (*field*), 945
sample_type (*field*), 990
sample_x (*field*), 550
sample_y (*field*), 550
sampled_fraction (*field*), 336
sas (*application definition*), *see* NXsas, 534
sastof (*application definition*), *see* NXsastof, 538
saturation_value (*field*), 242, 248, 522
scale (*field*), 1008
scaling (*field attribute*), 569
scaling_factor (*field attribute*), 325, 326, 438
scaling_factor (*field*), 186, 232
scan (*application definition*), *see* NXscan, 542
scan_number (*field*), 1052
scan_point_positions (*field*), 267
scan_schema (*field*), 388, 479
scanbox_em (*base class*), *see* NXscanbox_em, 387
scattering_angle (*field*), 1039
scattering_cross_section (*field*), 174
scattering_energy (*field*), 1039
scattering_length_density (*field*), 381, 386
scattering_vector (*field*), 206
scheme (*field*), 883, 924, 943, 944
Scientific Data Sets, *see* field
SDD (*field*), 442
SDS (*Scientific Data Sets*), *see* field
seblock (*field*), 544
seed (*field*), 694, 895
segment_columns (*field*), 207
segment_gap (*field*), 207
segment_height (*field*), 206
segment_rows (*field*), 207
segment_thickness (*field*), 206
segment_width (*field*), 206
semi_convergence_angle (*field*), 345
sensor (*base class*), *see* NXsensor, 389
sensor_material (*field*), 242, 522
sensor_scan (*application definition*), *see* NXsensor_scan, 1009
sensor_size (*field*), 429
sensor_thickness (*field*), 242, 522
separator (*base class*), *see* NXseparator, 1013
sequence_index (*field*), 340, 358, 614–618, 631–633, 934, 935
sequence_number (*field*), 238, 566
serial_number (*field*), 237, 287, 464–469, 609
set_Bfield_current (*field*), 1013, 1038
set_current (*field*), 917, 940, 1005, 1035
set_Efield_voltage (*field*), 1013, 1038
set_identifier (*field*), 656, 657
set_voltage (*field*), 917, 940
setpoint (*field*), 327–329, 353
setting (*field*), 476, 479
sgl (*field*), 552
sgu (*field*), 552
ShadowFactor (*field*), 441
shank_angle (*field*), 613
shape (*base class*), *see* NXshape, 392
shape (*field*), 171, 392, 441, 536, 539, 843, 860, 864, 868, 999, 1020, 1021, 1029, 1030
shermann_function (*field*), 1039
short_name (*field attribute*), 314, 394, 518, 527, 915
short_name (*field*), 169, 281, 389
short_title (*field*), 382
sigma_x (*field*), 396

sigma_y (field), 396
sign_convention (field), 459, 889
signal (field attribute), 231, 511, 575
signal (file attribute), 229
signal (group attribute), 243, 300, 434, 446, 469–475, 482, 483, 537, 618, 619, 744, 745, 898, 899, 945, 947
signal attribute value, 47, 229
signal data, 19, 47
signal_amplitude (field), 610
signal_type (field), 190
signed_distance (field), 760
silx (data analysis software), 1083
similarity_grouping (base class), *see* NXsimilarity_grouping, 1014
situation (field), 380, 427, 946
size (field), 171, 181, 392, 536, 539, 750, 843, 882, 999, 1020, 1021, 1029, 1030
slip_system_set (base class), *see* NXslip_system_set, 1016
slit (base class), *see* NXslit, 393
slit (field), 288
slit_angle (field), 253
slit_edges (field), 253
slit_height (field), 253
slit_length (field), 442
slits (field), 253
slot (field), 238
slow_axes (field), 916, 943
slow_pixel_direction (field), 251, 524
snapshot_x (field), 965
SNSbanking_file_name (field), 1018, 1027
SNSdetector_calibration_id (field), 1019, 1028
snsevent (application definition), *see* NXsnsevent, 1017
SNSgeometry_file_name (field), 1019, 1028
snshisto (application definition), *see* NXsnshisto, 1026
SNSmapping_file_name (field), 1018, 1027
SNStranslation_service (field), 1019, 1028
social_media_name (field), 603, 631, 712, 721, 730, 738, 778, 787, 793, 801, 812, 825, 951, 970
social_media_platform (field), 603, 631, 712, 721, 730, 738, 778, 787, 793, 801, 812, 825, 951, 970
soft_limit_max (field), 357
soft_limit_min (field), 357
software, 1081, 1082
solenoid_magnet (base class), *see* NXsolenoid_magnet, 1035
solid_angle (field), 237
solid_geometry (base class), *see* NXsolid_geometry, 1036
soller_angle (field), 194
source (base class), *see* NXsource, 394
source distribution, *see* NAPI installation
source_distance_x (field), 183
source_distance_y (field), 183
source_type (field), 921
space_group (field), 205, 381, 386, 417, 482
spatial_distribution_model (field), 962
spatial_filter (base class), *see* NXspatial_filter, 1036
spatial_resolution (field), 916
spe (application definition), *see* NXspe, 543
spec2nexus, 1084
special_enthalpy (field), 963
special_pre_factor (field), 963
spectral_range (field), 926
spectral_resolution (field), 838, 1045, 1046
spectrum (base class), *see* NXspectrum, 400
spectrum (field), 838, 1046
Sphinx (documentation generator), 1091
spin_rotator (base class), *see* NXspin_rotator, 1037
spindispersion (base class), *see* NXspindispersion, 1039
splitting_ratio (field), 842
spwidth (field), 419
sqom (application definition), *see* NXsqom, 545
stage_type (field), 986
standing_voltage (field), 610, 1002
start (field attribute), 238, 253, 283, 325, 326
start (field), 1007
start_time (field), 168, 213, 214, 238, 278, 285, 335, 409, 425, 429, 451, 454, 469, 505, 507, 513, 516, 530, 532, 535, 539, 542, 548, 551, 555, 557, 560, 562, 565, 570, 574, 601, 629, 633, 634, 712, 715, 716, 721, 724, 725, 729, 732, 733, 737, 760, 761, 777, 780, 781, 786, 788, 792, 795, 796, 800, 805, 806, 811, 818, 819, 825, 829, 830, 897, 934, 936, 942, 950, 969, 975, 976, 983, 1009, 1017, 1026, 1044, 1052
static_q_list (field), 1058
static_roi_map (field), 1058
status (field), 174, 182, 266, 290, 608, 630, 712, 721, 738, 777, 786, 793, 801, 811, 825, 969
status_indicators (field), 191
stepsize (field), 962
stop_time (field), 238
storage_mode (field attribute), 1053–1056
store_atomic_mass_sum (field), 682
store_charge_state (field), 683
store_disjoint_isotopes (field), 683
store_natural_abundance_product (field), 683
strategies, 70
 simplest case(s), 70
stress_field (field), 379, 427
stride (field), 1008
strings, 45
 arrays, 45
 fixed-length, 45

variable-length, 45
stxm (*application definition*), *see* NXstxm, 547
stxm_scan_type (*field*), 549
subentry
 NXsubentry, 161
subentry (*base class*), *see* NXsubentry, 407
subsampling_filter (*base class*), *see* NXsubsampling_filter, 1040
substance (*base class*), *see* NXsubstance, 1041
substrate (*field*), 990
substrate_density (*field*), 297, 331
substrate_material (*field*), 291, 297, 299, 331, 844, 938, 1000, 1050
substrate_roughness (*field*), 291, 297, 299, 331
substrate_thickness (*field*), 291, 297, 299, 331, 844, 938, 1000, 1050
support_membrane_material (*field*), 295
support_membrane_thickness (*field*), 295
surface (*field*), 300
surface_area (*field*), 848, 850, 860, 864, 869, 875, 876
surface_energy_density (*field*), 964
surface_indices (*group attribute*), 300
surface_type (*field*), 1003
symbol (*field attribute*), 185
symbols (*NXDL element*), 139
symbolsType (*NXDL data type*), 153
symmetric (*field*), 421
symmetry (*field*), 255, 631, 741, 855, 971

T

T (*field*), 447
T_axes (*group attribute*), 446
table (*field*), 420
tags, 131, 1078
taper (*field*), 313
target (*field*), 974, 1039
target, *attribute*, 20
target_dcom_radius (*field*), 672
target_edge_length (*field*), 672
target_material (*field*), 396
target_preparation (*field*), 1039
target_preparation_date (*field*), 1039
target_smoothing_step (*field*), 672
target_value (*field*), 357
targets (*field*), 714
tas (*application definition*), *see* NXtas, 551
Tdev (*field*), 447
technology (*field*), 634, 716, 724, 733, 761, 781, 788, 796, 805, 818, 830, 893, 976
telephone_number (*field*), 418, 456, 603, 631, 712, 721, 729, 738, 778, 787, 793, 801, 812, 825, 951, 970, 984, 1010, 1044
temperature (*field*), 207, 290, 334, 379, 427, 430, 445, 517, 544, 575, 611, 932, 946, 964, 973, 1020, 1029, 1059
temperature_coefficient (*field*), 207
temperature_range (*field*), 191
temperature_set (*field*), 1059
template, *see* NXDL template file
TERM (*field*), 186, 348
term (*field*), 445
text (*field*), 909
texture_index (*field*), 954
texture_strength (*field*), 954
thickness (*field*), 174, 206, 290, 293, 382, 444, 458, 837, 989
thickness_reduction (*field*), 935
thread_identifier (*field*), 816, 973, 974
three_dimensional_rotation_handedness (*field*), 951, 971
threshold (*field*), 694
threshold_distance (*field*), 650
threshold_energy (*field*), 242, 247, 522
threshold_proximity (*field*), 656
tiled
 tools, 1084
tilt1 (*field*), 319, 480
tilt2 (*field*), 319, 480
tilt_angle (*field*), 201, 582
tilt_correction (*field*), 346
time (*field attribute*), 174, 397
time (*field*), 264, 325, 866, 935, 953, 964, 973, 981, 1018, 1027
time_control (*field*), 934
time_of_flight (*field*), 235, 336, 511, 512, 533, 539, 540, 555, 556, 558, 560, 561, 1021, 1029, 1030
time_origin_location (*field attribute*), 1054
time_per_channel (*field*), 429, 519
time_points (*field*), 1045
time_stamp (*field*), 640, 649, 655, 661, 680, 687, 690, 698, 705, 710, 961
time_zone (*field*), 518
timestamp (*group attribute*), 435, 447
timing (*field*), 917, 940
Tischler, Jonathan, 11, 1089
title (*field*), 232, 268, 277, 398, 408, 425, 429, 433, 451, 469–475, 482, 483, 505, 507, 508, 513, 516, 530, 532, 535, 539, 542, 546, 548, 551, 555, 557, 560, 562, 565, 568, 570, 573, 574, 618, 619, 897, 942, 1017, 1026
tof_distance (*field*), 924
tofnpd (*application definition*), *see* NXtofnpd, 554
tofraw (*application definition*), *see* NXtofraw, 557
tofsingle (*application definition*), *see* NXtofsingle, 559
tolerance (*field*), 357
tomo (*application definition*), *see* NXtomo, 562
tomophase (*application definition*), *see* NXtomophase, 565

- t**
tomoproc (application definition), see NXtomoproc, **568**
top_dead_center (field), 253
top_up (field), **397**
total (field), 189, 751–753, **755**
total_area (field), **351**
total_counts (field), 1017, 1020, 1026, 1029
total_elapsed_time (field), 213, 633, 715, 724, 732, 760, 780, 788, 795, 805, 818, 829, 975
total_flux (field), **525**
total_flux_integrated (field), **526**
total_physical_memory (field), 634, 716, 724, 732, 761, 781, 788, 796, 805, 818, 830, 894, 975
total_uncounted_counts (field), 1017, 1026
transfer_rate (field), **926**
transform (field attribute), **569**
transformation_matrices, **25**
transformation_type (field attribute), **26**
transformation_type (field attribute), 250, 251, 413, 523, **524**
transformations (base class), see NXtransformations, **411**
translation, **26**
translation (base class), see NXtranslation, **414**
translation (field), **898**
transmission (application definition), see NXtransmission, **1043**
transmission (field), 444, 843, 938, **999**
transmittance (field), **993**
transmitting_material (field), 194, 289
TransportAppDef, **595**
tree structure, see hierarchy
triangle_cluster_identifier (field), **749**
triangle_identifier (field), **722**
trigger_dead_time (field), **241**
trigger_delay_time (field), **241**
trigger_delay_time_set (field), **241**
trigger_internal_delay_time (field), **241**
tutorial
 WONI, **58**
twist (field), **419**
two_time_corr_func (field), **1053**
type, see data type
type (field attribute), 341, 456, 457, **1041**
type (field), 169, 173, 174, 188, 194, 198, 205, 210, 211, 215, 234, 238, 252, 257, 258, 281, 286, 288, 293, 312, 317, 324, 327, 331, 333, 336, 340, 356, 371, 380, 390, 395, 419, 426, 429, 455, 459–461, 463, 464, 466, 468, 469, 473, 481, 482, 506, 509, 513, 522, 530, 535, 539, 546, 548, 563, 566, 568, 570, 574, 609, 694, 802, 835, 837, 841, 846, 895, 898, 925, 937, 943, 961, 998, 1019–1021, 1028–1030, 1039, 1046, 1049
type (group attribute), 279, **603**
- t**
type (NXDL attribute), **65**, **66**
type_dict_keyword (field), **958**
type_dict_value (field), **958**
type_other (field), **998**
- U**
- ub_matrix (field)*, **380**
UDunits, **46**
unassigned (field), **882**
uncertainties (field attribute), 436, 438, **447**
underload_value (field), 242, 248, **522**
Unidata UDunits, **46**
unit category, **156**
unit_cell (base class), see NXunit_cell, **415**
unit_cell (field), 206, 380, 511, 552, **575**
unit_cell_a (field), 206, 291
unit_cell_abc (field), 379, 386
unit_cell_alpha (field), 206, 291
unit_cell_alphabetagamma (field), 379, 386
unit_cell_b (field), 206, 291
unit_cell_beta (field), 206, 291
unit_cell_c (field), 206, 291
unit_cell_class (field), 381, 386
unit_cell_gamma (field), 206, 291
unit_cell_volume (field), 206, 291, 380, 386
units, 14, 18, 19, **46**, **133**
units (field attribute), 231, 348, 435, 438–440, 483, 836–838, 898, 899, 926, 985, 991, 992
units (NXDL attribute), **66**
upper_cap_radius (field), **848**
URL (field attribute), 278, 408
url (field attribute), 359, 909, 920, 983, **1043**
url (field), **192**, **1044**
url (group attribute), 985, 992, 1044
usage (field), **204**
use of, **161**
user (base class), see NXuser, **418**
UTF-8, **45**, **156**
utilities, **1081**
uuid (field), 210, 633, 715, 724, 732, 760, 780, 788, 795, 805, 818, 829, 975
- V**
- validation*, **73**, **1082**
validItemName (NXDL data type), **154**
validNXClassName (NXDL data type), **154**
validTargetName (NXDL data type), **155**
value (field), 281, 325, 327–329, 347, 354, 357, 390, 609, 907, 910–912, 918, 940, 974, 975, 1005, 1011, 1013, 1018, 1020, 1021, 1027, 1029, 1030, 1035, 1038
value (NXDL attribute), **66**
value (transformation matrix), **26**
value_deriv1 (field), **390**

value_deriv2 (*field*), 390
value_timestamp (*field*), 1011
value_x (*field*), 258, 478
value_y (*field*), 258, 478
values (*field*), 905, 906, 910–912, 987
variable (*field*), 509
varied_parameter_link (*field*), 992
varied_variable (*field attribute*), 509
vector (*field attribute*), 26, 179, 180, 250, 251, 413, 524
vector (*field*), 899
vector (*group attribute*), 900
velocity (*field*), 357
velocity_selector (*base class*), *see* NXvelocity_selector, 419
vendor (*field*), 287, 464–469, 608–612, 934, 937
verification, 73
version (*field attribute*), 210, 277, 278, 287, 359, 408, 409, 425, 455, 464, 480, 544, 602, 614–620, 630, 633, 638–643, 648, 649, 651, 654, 657, 658, 660–663, 670, 672, 674, 680, 681, 683, 686, 687, 690, 692, 698, 699, 704–706, 709–712, 715, 720, 721, 724, 729, 732, 737, 760, 776, 777, 780, 786, 788, 792, 795, 800, 805, 811, 818, 824, 825, 829, 862, 895, 908, 919, 942, 950, 961, 962, 969, 970, 975, 983, 984, 1009, 1010, 1043
version (*field*), 358, 509, 546, 568, 573, 909, 985, 992, 993, 1019, 1027
version (*group attribute*), 433, 516, 601, 629, 640, 648, 654, 660, 680, 686, 690, 698, 704, 709, 711, 720, 728, 737, 776, 785, 792, 800, 811, 824, 933, 936, 949, 961, 968, 985
vertex_identifier (*field*), 852
vertex_identifier_offset (*field*), 743, 746, 756, 758, 804, 817, 851, 857
vertex_incident_half_edge (*field*), 857
vertices (*field*), 216, 344, 688, 743, 747, 750–752, 754, 755, 757, 758, 804, 817, 852, 872
vertices_are_unique (*field*), 853, 872
virtual_pixel_interpolation_applied (*field*), 241, 521
vocabulary, 160
voltage (*field*), 234, 257, 258, 303, 324, 396, 476, 479, 932
volume (*field*), 173, 417, 750–754, 803, 804, 816, 848, 850, 860, 869, 875, 876, 974
volume_fraction (*field*), 381, 386, 954
voxel_identifier (*field*), 632
vscode-h5web
 tools, 1084

W

warmup (*field*), 694, 895

wavelength (*field*), 206, 243, 288, 300, 338, 397, 420, 506, 513, 530, 535, 575, 581, 610, 907, 910, 911, 1002, 1021, 1030, 1045
wavelength_dispersion (*field*), 338
wavelength_error (*field*), 338
wavelength_errors (*field*), 338
wavelength_identifier (*field*), 905, 910–912
wavelength_indices (*group attribute*), 243, 300
wavelength_max (*field*), 444, 905
wavelength_min (*field*), 444, 905
wavelength_range (*field*), 254, 842, 998, 1046
wavelength_range_coating (*field*), 844, 1050
wavelength_spread (*field*), 420, 535
wavelength_unit (*field*), 905, 910–912
wavelengths (*field*), 1049
waveplate (*base class*), *see* NXwaveplate, 1048
webpage, 1073
wedge_angle (*field*), 843, 999
weight (*field*), 273, 714, 740, 903
weighting_model (*field*), 739, 902
weinberg_vector (*field*), 858
why NeXus?, *see* motivation, 11
width (*field*), 288, 420, 860, 864
winding_order (*field*), 344, 853
window_effects_corrected (*field*), 988
windowing_method (*field*), 643, 649, 661, 681, 687, 692, 699, 705, 1037
Windows, *see* NAPI installation
WONI, 58
workflow_description (*field*), 950
workflow_identifier (*field*), 950
workflow_step_description (*field*), 933
workflow_step_identifier (*field*), 933
working_distance (*field*), 188, 345, 883
write file, 13

X

x (*field*), 181, 199, 232, 459, 460, 462, 463, 481, 569
x_alias (*field*), 199
x_boundary_convention (*field*), 264, 481
x_direction (*field*), 199, 263, 459, 460, 462, 463, 481
x_gap (*field*), 394, 441
x_indices (*field*), 899
x_normalization_direction (*field*), 264, 481
x_pixel_offset (*field*), 236, 1020, 1029
x_pixel_size (*field*), 237, 443, 511, 533, 536, 540, 563, 566, 575, 898, 1057
x_position (*field*), 442, 445
x_rotation_axis_pixel_position (*field*), 563
x_translation (*field*), 382, 563, 566, 575
xas (*application definition*), *see* NXxas, 570
xasproc (*application definition*), *see* NXxasproc, 572
xaxis_alias (*field*), 952, 971
xaxis_direction (*field*), 952, 971

xbase (*application definition*), *see* NXxbase, **574**
xdmf_cell_identifier (*field*), **817**
xdmf_feature_identifier (*field*), **750**, **751**, **753–755**
xdmf_gradient (*field*), **745**
xdmf_intensity (*field*), **744**
xdmf_ion_identifier (*field*), **722**
xdmf_topology (*field*), **617**, **744**, **745**, **747**, **750**, **751**,
753–755, **757**, **758**, **804**, **817**, **826**
xdmf_xyz (*field*), **744**, **745**
xeuler (*application definition*), *see* NXxeuler, **577**
xkappa (*application definition*), *see* NXxkappa, **578**
xlaue (*application definition*), *see* NXxlaue, **580**
xlaueplate (*application definition*), *see* NXxlaueplate,
581
XML, **1088**
XML_version (*file attribute*), **374**
xmlns (*NXDL attribute*), **65**
xnb (*application definition*), *see* NXxnb, **582**
xpcs (*application definition*), *see* NXxpcs, **1051**
xpos (*field*), **744**, **745**
xpos_indices (*group attribute*), **744**, **745**
xraylens (*base class*), *see* NXxraylens, **421**
xrot (*application definition*), *see* NXxrot, **583**
xsi:schemaLocation (*NXDL attribute*), **65**

Y

y (*field*), **181**, **199**, **233**, **459**, **460**, **462**, **464**, **481**, **569**
y_alias (*field*), **199**
y_boundary_convention (*field*), **264**, **481**
y_direction (*field*), **199**, **263**, **459**, **460**, **462**, **464**, **481**
y_gap (*field*), **394**, **441**, **1045**
y_indices (*field*), **899**
y_normalization_direction (*field*), **264**, **481**
y_pixel_offset (*field*), **236**, **1020**, **1029**
y_pixel_size (*field*), **237**, **443**, **511**, **533**, **536**, **540**, **563**,
566, **575**, **899**, **1057**
y_position (*field*), **442**, **445**
y_rotation_axis_pixel_position (*field*), **563**
y_translation (*field*), **563**, **566**, **575**
yaw (*field*), **442**, **445**
yaxis_alias (*field*), **952**, **971**
yaxis_direction (*field*), **952**, **971**
ypos (*field*), **744**, **745**
ypos_indices (*group attribute*), **744**, **745**

Z

z (*field*), **199**, **233**, **459**, **461**, **462**, **464**, **481**, **569**
z_alias (*field*), **199**
z_direction (*field*), **199**, **263**, **459**, **461**, **462**, **464**, **481**
z_pixel_offset (*field*), **236**
z_translation (*field*), **563**, **566**
zaxis_alias (*field*), **952**, **971**
zaxis_direction (*field*), **952**, **971**
zone_height (*field*), **294**